



EFW

I hereby certify that this paper (along with any paper referred to as being attached below with sufficient postage as First Class Mail, in an envelope addressed to: MS Amendment, Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

Dated: April 30, 2007

Signature: 

(Raymond B. Churchill)

Docket No.: SCEI 3.0-155
(PATENT)

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Patent Application of:
Yamazaki et al.

Application No.: 10/725,129

Group Art Unit: 2185

Filed: December 1, 2003

Examiner: Not Yet
Assigned

For: METHODS AND APPARATUS FOR
EFFICIENT MULTI-TASKING

MS Amendment
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

**INFORMATION DISCLOSURE STATEMENT AND
CERTIFICATION PURSUANT TO 37 C.F.R. § 1.97(E) (1)**

Dear Sir:

It is respectfully requested that the reference listed on the enclosed form be made of record and considered with respect to the above-referenced U.S. patent application. The reference was cited in an office action issued in connection with the applicants' corresponding Japanese application. A copy of the reference and the office action are enclosed. Submission of the present Information Disclosure Statement should not be taken as an admission that the cited reference is legally available prior art or that the same is pertinent or material.

Pursuant to 37 C.F.R. § 1.97(e) (1), undersigned counsel hereby certifies that each item of information contained in this Information Disclosure Statement was first cited in a communication from a foreign patent office in a counterpart foreign application to the above-referenced patent application not more than three months prior to the filing of said statement.

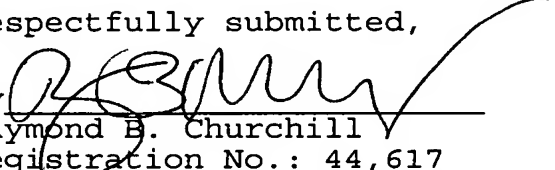
Application No.: 10/725,129

Docket No.: SCEI 3.0-155

In the event that any fee is due in connection with the present Information Disclosure Statement, the Commissioner is hereby authorized to charge the same to our Deposit Account No. 12-1095.

Dated: April 30, 2007

Respectfully submitted,

By 
Raymond B. Churchill
Registration No.: 44,617
LERNER, DAVID, LITTENBERG,
KRUMHOLZ & MENTLIK, LLP
600 South Avenue West
Westfield, New Jersey 07090
(908) 654-5000
Attorney for Applicant(s)

LD-458\



PTO/SB/08A/B (09-06)

Approved for use through 03/31/2007. OMB 0651-0031

U.S. Patent and Trademark Office; U.S. DEPARTMENT OF COMMERCE

Under the Paperwork Reduction Act of 1995, no persons are required to respond to a collection of information unless it contains a valid OMB control number.

Substitute for form 1449/PTO

**INFORMATION DISCLOSURE
STATEMENT BY APPLICANT**

(Use as many sheets as necessary)

Sheet 1 of 1

Complete if Known

Application Number	10/725,129-Conf. #2626
Filing Date	December 1, 2003
First Named Inventor	Takeshi Yamazaki
Art Unit	2185
Examiner Name	Not Yet Assigned
Attorney Docket Number	SCEI 3.0-155

U.S. PATENT DOCUMENTS

Examiner Initials*	Cite No. ¹	Document Number Number-Kind Code ² (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear

FOREIGN PATENT DOCUMENTS

Examiner Initials*	Cite No. ¹	Foreign Patent Document Country Code ³ -Number ⁴ -Kind Code ⁵ (if known)	Publication Date MM-DD-YYYY	Name of Patentee or Applicant of Cited Document	Pages, Columns, Lines, Where Relevant Passages or Relevant Figures Appear	T ⁶
	BA	JP-06-222936	08-12-1994	lbn		**

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant. ¹ Applicant's unique citation designation number (optional). ² See Kinds Codes of USPTO Patent Documents at www.uspto.gov or MPEP 901.04. ³ Enter Office that issued the document, by the two-letter code (WIPO Standard ST.3). ⁴ For Japanese patent documents, the indication of the year of the reign of the Emperor must precede the serial number of the patent document. ⁵ Kind of document by the appropriate symbols as indicated on the document under WIPO Standard ST.16 if possible. ⁶ Applicant is to place a check mark here if English language Translation is attached.

NON PATENT LITERATURE DOCUMENTS

Examiner Initials*	Cite No. ¹	Include name of the author (in CAPITAL LETTERS), title of the article (when appropriate), title of the item (book, magazine, journal, serial, symposium, catalog, etc.), date, page(s), volume-issue number(s), publisher, city and/or country where published.	T ²

*EXAMINER: Initial if reference considered, whether or not citation is in conformance with MPEP 609. Draw line through citation if not in conformance and not considered. Include copy of this form with next communication to applicant.

¹ Applicant's unique citation designation number (optional). ² Applicant is to place a check mark here if English language Translation is attached.

** - with English abstract.

Examiner Signature		Date Considered	
--------------------	--	-----------------	--

PATENT ABSTRACTS OF JAPAN

(11)Publication number : 06-222936

(43)Date of publication of application : 12.08.1994

(51)Int.Cl.

G06F 9/46

G06F 15/16

(21)Application number : 05-315383

(71)Applicant : INTERNATL BUSINESS MACH CORP <IBM>

(22)Date of filing : 15.12.1993

(72)Inventor : STONE HAROLD S
JANIS MURPHY STONE

(30)Priority

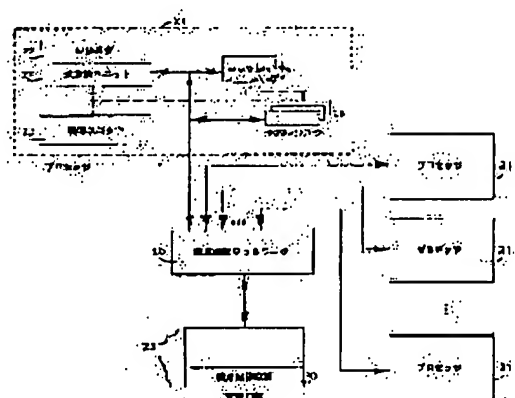
Priority number : 92 993193 Priority date : 18.12.1992 Priority country : US

(54) METHOD FOR UPDATING VALUE OF SHARED VARIABLE

(57)Abstract:

PURPOSE: To provide an improved method for atomically updating a variable shared by plural programs or processors by using reservation.

CONSTITUTION: Each computer processor 21 operating in a multiprocessing environment includes plural reservation registers 26 for setting reservation for each of plural shared variables, and the plural shared variables can be reserved during the execution of a program. The reservation registers 26 record the addresses of the reserved shared variables, the values of corrected results to be updated in the shared variable addresses, whether or not the variables are updated, or whether or not the reservation is valid. When an instruction for updating the shared variables is generated, the variable having the non- updated corrected result of a pair of reserved corrected shared variables designated by the instruction is updated as long as all the reservations are valid.



LEGAL STATUS

[Date of request for examination] 15.12.1993

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number] 2500101

[Date of registration] 01.03.1998

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right] 01.03.1999

(19)日本国特許庁(JP)

(12)公開特許公報(A)

(11)特許出願公開番号

特開平6-222936

(43)公開日 平成6年(1994)8月12日

(51)Int.Cl.³G 0 6 F 9/46
15/16

識別記号

3 4 0 F 8120-5B
3 5 0 R 7429-5L

庁内整理番号

FI

技術表示箇所

審査請求 有 請求項の数12 OL (全 24 頁)

(21)出願番号 特願平5-315383

(22)出願日 平成5年(1993)12月15日

(31)優先権主張番号 993193

(32)優先日 1992年12月18日

(33)優先権主張国 米国(US)

(71)出願人 390009531

インターナショナル・ビジネス・マシー
ズ・コーポレーションINTERNATIONAL BUSIN
ESS MACHINES CORPO
RATIONアメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 ハロルド・スチュアート・ストーン

アメリカ合衆国 ニューヨーク州チャパ
ククロス・リッジ・ロード 50

(74)代理人 弁理士 合田 深 (外3名)

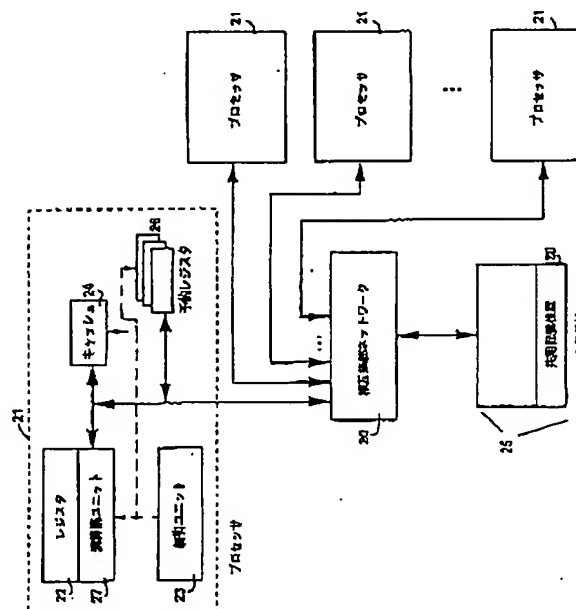
最終頁に続く

(54)【発明の名称】 共用変数の値を更新する方法

(57)【要約】

【目的】 本発明の目的は、予約を使用して、複数のプログラムまたはプロセッサが共用する変数を原子的に更新する、改良された方法を提供することである。

【構成】 マルチプロセッシング環境で動作する各コンピュータ・プロセッサは、複数の共用変数のそれぞれに対する予約を置くために複数の予約レジスタを含み、それらによってプログラムの実行中に複数の共用変数をそれぞれ予約することができる。予約レジスタは、予約中の共用変数のアドレス、共用変数アドレス中の更新される修正結果の値、その変数が更新されたかどうか、その予約が有効であるかどうか等を記録する。共用変数を更新しようとする命令に出会ったとき、その命令によって指定される予約された修正済みの1組の共用変数のうちで未更新の修正結果を有する変数が、そのすべての予約が有効である場合に限り更新される。



(2)

特開平6-222936

1

2

【特許請求の範囲】

【請求項1】 a. 原子的動作が依存する変数とその原子的動作によって変更される変数とを含む1組の共用変数中の指定された複数の共用変数のそれぞれについて、プロセスによって予約を置き、予約された共用変数のそれぞれに別々の予約記憶位置を関連付けるステップと、
b. 予約された共用変数が異なるプロセスによって修正される時に、予約された変数の予約を無効化するステップと、

c. 前記1組中の共用変数の値を更新する前に、前記1組中のすべての予約の有効性を検査するステップと、
d. 前記1組中のすべての予約の状況が有効である場合に限って、前記1組中のすべての修正済みの予約された共用変数の位置の値を原子的に更新するステップとを含む、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項2】 前記1組中の1つまたは複数の予約が無効である場合に、共用変数が更新されないことを特徴とする、請求項1に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項3】 予約された共用変数が、異なるプロセスによって、その値の修正なしにアクセスされる時にも、予約された共用変数の予約が無効化されることを特徴とする、請求項1に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項4】 指定された1組中の共用変数のそれぞれが、マルチプロセッシング環境内の、その共用変数を更新する特権を有するプロセッサによってのみ更新されることを特徴とする、請求項1に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項5】 a. LR（予約付きロード）命令を使用して、複数の共用変数のプロセスによる値を読み取り、読み取ったそれぞれの値のアドレスを、LR命令を実行するプロセッサの関連アドレス記憶位置に記憶することによって、読み取った各共用変数に対する予約を置くステップと、

b. 1つまたは複数の読み取った値を修正して、読み取ったそれぞれの値の修正結果を得るステップと、

c. SC（条件記憶）命令を使用して、結果記憶位置に各修正結果を記憶し、SC命令が、予約が有効であることを示すため、修正結果を有する予約された共用変数に関連する有効性記憶位置をも更新するステップと、

d. 異なるプロセスによって修正された共用変数の有効性記憶位置を無効化するステップと、

e. プロセス内で必要な1組の共用変数を識別するステップと、

f. 前記1組中のすべての共用変数の予約が有効である場合に限って、WR命令を使用して、前記1組の共用変数のそれぞれの修正結果を、それらの共用変数アドレスに原子的に書き戻すステップとを含む、マルチプロセ

シング環境で共用変数の値を更新する方法。

【請求項6】 1つまたは複数の予約が無効であると、前記1組中の共用変数がまったく更新されず、ステップaないしfの再実行が可能になることを特徴とする、請求項5に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項7】 共用変数が、その共用変数を更新する特権を有するプロセスによってのみ更新されることを特徴とする、請求項5に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項8】 アドレス記憶位置および結果記憶位置が、予約される共用変数に関連する予約レジスタ上のフィールドであることを特徴とする、請求項5に記載の、マルチプロセッシング環境で共用変数の値を更新する方法。

【請求項9】 相互接続ネットワークによって、共用記憶位置を有する主記憶域に接続されたプロセッサと、結果を計算するための各プロセッサ内の演算ユニットと、

計算された結果を記憶するための、各プロセッサ内のプロセッサ・レジスタのバンクと、

共用記憶位置を予約するための、プロセッサのそれぞれに含まれる複数の予約レジスタのバンクと、

演算ユニットとプロセッサ・レジスタと予約レジスタとを制御するための制御論理機構を有する、各プロセッサ内の制御ユニットと、

共用記憶位置のコピーを記憶するためのキャッシュ記憶域とを備える、マルチプロセッシング環境内のコンピュータのシステム。

【請求項10】 各予約レジスタが、さらに、

予約レジスタによって予約される共用記憶位置のアドレスを記憶するためのアドレス・フィールドと、

共用記憶位置の予約の有効性を示す情報を記憶するための有効性フィールドと、

予約レジスタの常駐するプロセッサが、アドレス・フィールドにそのアドレスが記憶された共用記憶位置を更新する特権を有するかどうかを示す情報を記憶するための特権フィールドと、

共用記憶位置を更新する修正結果を記憶するためのデータ・フィールドとを備える、請求項9に記載の、マルチプロセッシング環境内の1つまたは複数のコンピュータのシステム。

【請求項11】 複数のプロセスによる修正の対象となり得る変数にアクセスする、ユニプロセッサ・プログラム内の1組の書き込み命令と読取り命令を識別するステップと、

読み取られた各値のアドレスを、LR（予約付きロード）命令を実行するプロセスの関連アドレス記憶位置に記憶することによって、各読取り命令を、読み取られた各共用変数の値に対する予約を置く、LR命令に変更するステップと、

(3)

特開平6-222936

3

4

各書き込み命令を、記憶位置の修正結果を結果記憶位置に記憶し、修正結果を有する各共用変数に関連する有効性記憶位置を共用変数に対する予約が有効であることを示すように更新するSC（条件記憶）命令に変更するステップと、

異なるプロセスによって修正される共用変数の有効性記憶位置を無効化するステップと、

LR命令によって予約されるアドレス記憶位置を指定し、前記1組中のすべての共用変数の予約が有効である場合に限って、前記1組の共用変数のそれぞれの修正結果を当該のそれぞれの共用変数アドレスに書き戻し、そうでない場合には、どの共用変数をも更新しない、WR（予約時書き込み）命令を、プログラム中の最後のSC命令の直後に挿入するステップと、

WR命令が共用変数を更新しなかった場合に実行される、最初のWR命令に戻る条件付き分岐を挿入するステップとを含む、ユニプロセッサ上で走行するように設計されたコンピュータ・プログラムを、マルチプロセッシング計算環境で走行するコンピュータ・プログラムに変換する方法。

【請求項12】各共用変数に関連するアドレス記憶位置および結果記憶位置がそれぞれ、予約される共用変数に関連する予約レジスタ内のフィールドであることを特徴とする、請求項11に記載の、ユニプロセッサ・プログラムをマルチプロセッサ環境で走行するように変換する方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】本発明は、マルチプロセッシング・コンピュータ環境での共用変数の更新の分野に関する。具体的に言うと、本発明は、複数の予約レジスタによって複数の共用変数の原子的更新を実行する能力に関する。

【0002】

【従来の技術】現代のマルチプロセッサ・コンピュータ・システムは、複数のプロセッサが共通記憶位置にアクセスし、それを修正する手段を提供する。このような位置を「共用」位置と称し、各位置が共用変数を含むと称する。

【0003】コンピュータ・システム内に共用変数が存在するもう1つの状況は、別々の異なるプログラム

（「プロセス」と称することもある）が単一のプロセッサ内に同時に常駐するシングル・プロセッサ内である。各プロセスは、「量子」と称する時間の間制御を得、その量子の終わりにそのプロセスが中断されて、別のプロセスがプロセッサへのアクセスを許される。プロセスの実行は、当該のそれぞれの時間量子が、時間的にインターレースされ、ラウンド・ロビン方式または優先順位に基づいて許可される形になる。プロセスは、共通位置にアクセスしそれを修正することができ、したがって、シ

ングル・プロセッサ内のこれらのプロセスは、論理的には、別々の異なるのプロセッサ上で実行しているかのように共用記憶位置に作用する。

【0004】どちらの場合でも、複数の並行プロセスが共用変数を更新できるので、矛盾した更新が行われる可能性がある。2つのプロセスにそれぞれ、ある記憶位置を増分するタスクが割り当てられていると仮定する。各プロセスは、その位置から計算機レジスタに読み取り、計算機レジスタを増分し、次いで計算機レジスタの新しい値を記憶域に書き込むことによって、この作業を行おうとする。プロセス1が、プロセス2からの介入なしに読み取り／修正／書き込み（Read/Modify/Write）動作を実行し、その後プロセス2が同一の動作を実行する場合、この共用変数は、2回増分されることによって正しく修正される。

【0005】2つのプロセスの読み取り／変更／書き込み動作が時間的にインターレースされる時、矛盾が発生する。すなわち、プロセス1が変数Xの値を読み取り、次にプロセス2がXの同じ値を読み取り、その後、後続の増分と書き込みが任意の順序で行われる場合、各プロセスは、Xの元の値を1回増分した値に等しい値を記憶する。したがって、Xの最終的な値は、1回の増分しか反映しないことになり、1回の増分が失われる。

【0006】不正な最終値が得られる理由は、2つのプロセスが矛盾したデータに作用することである。プロセス1が、修正する意図でXの値を読み取る場合、Xの論理常駐は、共用記憶域からプロセス1に関連する私用レジスタに移動する。その後、プロセス2がXを読み取り、Xがまだプロセス1によって修正されていない場合、プロセス2の読み取る値は、Xの実際の値と矛盾する。というのは、Xは、現在共用記憶域内でアクセスできず、再書き込みされるまでアクセス可能にならないからである。

【0007】正しさを保証するには、整合性を確保する必要がある。整合性のある挙動のための十分条件は、読み取り／変更／書き込み動作シーケンスが共用変数Xに対して実行される時に、そのシーケンスが原子的に実行されることを保証することである。原子的とは、本明細書では、Xの読み取りが実行されてからXの書き込みが実行されるまでの間に他のプロセスがXの値を変更することが許されずに、そのシーケンスが実行されることを意味する。

【0008】読み取り／変更／書き込み動作シーケンスの正しさと整合性を保証する方法は、多数存在する。その1つは、そのようなシーケンスをクリティカル・セクションと称するプログラム領域に囲い込むことである。クリティカル・セクションの前には、ロック動作を実行する1つまたは複数の命令があり、クリティカル・セクションの後には、ロック解除動作を実行する1つまたは複数の命令がある。このロック動作とロック解除動作によっ

(4)

特開平6-222936

5

て、クリティカル・セクション内で、一時にせいぜい1つのプロセスしかコードを実行できないようになる。規則またはプロトコルにより、他のすべてのプロセスが、同じロック/ロック解除機構によって制御されるクリティカル・セクションによって同じ共用変数を更新する場合には、クリティカル・セクション内の読取り/変更/書き込み動作によって、共用変数を原子的に更新することができる。というのは、あるプロセスがクリティカル・セクション内で読取り/変更/書き込み処理を実行している間、他のプロセスが共用変数にアクセスできないからである。

【0009】クリティカル・セクションを作成するには、ロック機能を有することが必要である。従来技術では、このような機能を実施するための一般的な手段は、それ自体がロック変数に対して原子的読取り/変更/書き込み動作を実行する1つの命令によるものである。この目的に使用される命令には、テスト後セット (Test-and-Set) 命令と、増分 (Increment) (または減分 (Decrement)) 命令がある。

【0010】テスト後セット命令は、変数Xを読み取り、Xのうちの1ビットを、そのビットの元の状態と無関係に値1にセットし、Xを再書き込みし、値1にセットされる前の元のビットに等しい値の条件コードを返す。この命令は、共用変数の読取り/変更/書き込み動作を実行するので、実行の整合性を保証するために、そのような更新を原子的に実行するように実施しなければならない。この1つの命令を使用して、任意の複雑さの読取り/変更/書き込み動作シーケンスを矛盾なしに実行できるクリティカル・セクションを保護するロックをセットすることができる。というのは、このロックによって、他のプロセスがその間に共用変数にアクセスできなくなるからである。

【0011】テスト後セット命令を使用してロックを作成するために、各プロセスが、ロック変数にアクセスするテスト後セット命令で、クリティカル・セクションを保護する。このテスト後セット命令の実行の直後に、ロック変数は、前の値と無関係に1の値を保持する。各プロセスは、テスト後セット命令の実行後に条件コードをテストし、0 (初期値の0に対応する) のコードを観察したプロセスだけが、クリティカル・セクションに入り、クリティカル・セクション内で読取り/変更/書き込み動作シーケンスを実行することができる。それぞれのクリティカル・セクションで読取り/変更/書き込みコードを実行しようと試みる他のすべてのプロセスは、クリティカル・セクションのロックがロック解除 (0) 状態になるまで、待つかあるいは他の動作を行わなければならない。あるプロセスが読取り/変更/書き込み動作シーケンスを完了し、クリティカル・セクションから離れる時、そのプロセスは、ロックに0を記憶することによ

6

てロックをクリアし、それによって、他のプロセスが、変数の初期値が0に等しいロックに対してテスト後セット命令を実行することによって、クリティカル・セクションに入ることができるようになる。

【0012】テスト後セット命令は、IBM 370にある。増分命令と減分命令は、原子的読取り/変更/書き込み動作として実施することができ、ほとんどテスト後セット命令の直接の代替物として、テスト後セット命令の行うのとほぼ同じことを実行するのに使用できる。テスト後セット命令は1ビットしかセットできないが、増分命令と減分命令は、共用変数を原子的に増分または減分できるので、融通性がより高い。DEC VAXは、原子的な増分命令および減分命令を有する。

【0013】読取り/変更/書き込み動作を使用してクリティカル・セクションを作成することには、幾つかの問題がある。これには、下記のものが含まれる。

1. クリティカル・セクションには、一時にせいぜい1つのプロセスしか入ることができない。あるプロセスがクリティカル・セクションに入って障害を発生した場合、他のプロセスがクリティカル・セクションに入れなくなり、システム全体が障害を発生する。

2. 長く複雑なクリティカル・セクションは、マルチプロセッサ・システムの性能上のネックになる。一時に1つのプロセスしかクリティカル・セクションに入ることができないので、他のプロセスが同じクリティカル・セクションに入らなければならない場合、それらのプロセスは、クリティカル・セクションを使用できるようになるまで待たなければならない。この時間の間強制的に遊休状態になる可能性がある。この問題に対する望ましい解決策は、クリティカル・セクションの外部で読取り/変更/書き込み動作を実行する命令シーケンスによって共用変数を更新することである。この修正は、整合性が保たれるように注意深く制御しなければならない。

3. クリティカル・セクションには多くの命令が含まれるので、あるプロセスの時間量子があるクリティカル・セクション内で終了する可能性があり、他のプロセスがプロセッサの制御を得ている間、そのプロセスが長時間にわたって中断される可能性がある。クリティカル・セクションのロックを保持するプロセスが中断されている間、他のプロセスは、クリティカル・セクションに入ることができず、そのクリティカル・セクションによって制御される共用変数を更新することができない。

【0014】これらの欠点を克服するため、共用変数に対する複雑な動作を、原子的に実行される単一命令として実施することが可能である。1例として、DEC VAXは、それぞれ最高4つの共用変数を同時に原子的に変更し、したがって矛盾なしに変更を行う、原子的な待機 (ENQUEUE) 命令および待機解除 (DEQUEUE) 命令を有する。

【0015】この手法に従う場合、計算機設計者は、各

(5)

特開平6-222936

7

データ表現ごとに、共用変数に対する異なる各動作ごとに原子的命令を供給しなければならない。たとえば、DEC VAXは、異なる表現を有する待ち行列用の異なる待機命令および待機解除命令を有する。要素ごとに1つのポインタを有する待ち行列は、待機/待機解除命令対のうちの1つによって操作しなければならない。要素ごとに2つのポインタを有する待ち行列は、異なる待機/待機解除命令対によって操作しなければならない。

【0016】この手法の問題点は、計算機設計者が、考えられるすべての原子的動作および考えられるすべてのデータ表現を予測しなければならないことである。というのは、それらの動作がそれぞれ、極度に特殊化されているからである。プログラマが、特定の各文脈および各データ表現用の専用の原子的動作を作成し、使用することは簡単ではないので、この手法は失敗する。

【0017】したがって、対処すべき重要な問題は、共用変数に対する複雑な更新のためのカスタマイズされた原子的動作を生成し、これらの原子的動作を実行して、単一命令の実行中に実際の更新が行われるようにする能力である。このようにすると、マルチプロセッサ・システムの信頼性と性能が、クリティカル・セクションのロックを保持するプロセスの障害または割込みのために危険にさらされることがなくなる。

【0018】この問題に対する部分的な解決策は、IBM 370上で実施されている比較交換 (Compare-and-Swap) 命令を使用して得ることができる。比較交換命令は、クリティカル・セクションによって保護されない読取り/変更/書き込み命令シーケンスの最終ステップを提供し、整合性と原子性が保証されると思われる特定の条件の下でのみ書き込み動作を実行する。比較交換命令を使用するには、プロセスは、まず従来の非原子的命令によって、共用変数更新の読取り/変更/書き込み動作シーケンスのうち、最後の共用変数の更新済み値の書き込み以外をすべて実行する。最後の書き込み動作を実行する際に、プロセスは、比較交換命令を使用する。比較交換命令は、記憶域から変数を再読取りし、記憶域内の現在値を、計算機レジスタ内に保持された前の値のコピーと比較する。この2つの値が同一である場合、比較交換命令は、記憶域内の変数を、新たに計算された値で更新する。計算機レジスタ内に保持された前の値のコピーと、共用記憶域に保持される現在の値が異なる場合、比較交換命令は、記憶域を更新せず、計算機レジスタ内に保持された前の値のコピーを、共用記憶域内の変数の現在値のコピーで置き換える。この命令は、何が起こったかを条件コードで報告する。比較交換命令自体が読取り/変更/書き込み動作シーケンスを実行するので、プログラムの正しさを保証するために、これらの動作を原子的に実行しなければならない。

【0019】比較交換の概念は、共用値が最後に読み取られて以降に、他のプロセスがその値を変更していない

8

ならば、その共用値を更新できるというものである。これは、整合性のある挙動を保証すると思われる。困ったことに、矛盾した更新の可能性がある。第1のプロセスが、比較交換命令で終了する読取り/変更/書き込み動作シーケンスを実行している最中に、並行して実行中の第2のプロセスが、その変数の値 (たとえば値A) を読み取り、その値を新しい値 (たとえば値B) に変更し、さらに別のプロセスが、第1のプロセスが値Aに基づいて新しい値を計算している間に、この共用変数を前の値 (値A) に戻すことがあり得る。第1のプロセスが比較交換命令に達した時、現在値と前の値はどちらもAに等しいので、この比較交換命令は成功する。

【0020】しかし、多くの応用例では、正しさが成り立つためには、共用変数の前の値と現在値が等しいだけでなく、記憶域内の変数が、読取り/変更/書き込み動作シーケンスの実行中に初めて読み取られてから、比較交換命令が実行されるまでの間、連続的に値Aを維持することが必要である。比較交換命令がこのような値の変化を検出できないことを、「ABA問題」と称する。

【0021】共用変数の並行アクセスおよび並行修正用のアルゴリズムのほとんどが、ABA問題が発生し得る場合に障害が発生する可能性がある。ABA問題に起因する障害の可能性を除去または低下させるために比較交換命令と共に使用されるさまざまなソフトウェア方式が存在する。

【0022】この問題に対するもう1つの手法は、比較交換命令の使用に非常に類似しているが、更新を試みる時刻に変数がある値を連続して維持しているかどうかを検出する点異なる。要するに、この方式は、比較交換命令が、現在値と前の値が等しいかどうかテストするのではなく、変数が変化したかどうかテストするかのように動作する。この方式では、「予約」と称する概念を使用する。

【0023】この方式では、予約付きロード (Load-with-Reservation, LR) 命令と予約時書き込み (Write-if-Reserved, WR) 命令という2つの命令を使用する。ただし、プロセッサによって命令の名称は異なる。これは、MIPS R4000プロセッサ [Kane, 1989] およびDEC ALPHAプロセッサ [Digital Equipment Corporation, 1992] で現在使用されている。LR命令は、記憶域から共用変数を読み取り、それと同時に、その変数のアドレスを「予約レジスタ」と称する特殊レジスタに置く。予約レジスタは、他のプロセスによって加えられるその変数に対する変更を監視する。他のプロセスとは、同一のプロセッサ上または他のプロセッサ上で実行中のプロセスである。他のプロセスがこの変数の値を変更する場合、予約が消滅する。そうでない場合、予約を置いたプロセスがWR命令に達するまで、予約が存在し続ける。WR命令は、予約がまだ存在している場合

(6)

特開平6-222936

10

9

に限って更新を実行する。そうでない場合、WR命令は更新を実行しない。どちらの場合でも、この命令は、何が起こったかを示す条件コードを返す。多くの応用例では、このプロセスは、その後、条件コードをテストし、更新が完了するまで、LR、修正、WRのシーケンスを繰り返す。

【0024】従来技術の問題点

予約という概念を用いると、単一の変数を更新する場合、上記の問題のほとんどが解決される。しかし、共用データ構造に作用するプロセスのほとんどが、複数の共用変数に原子的に作用しなければならない。すなわち、データ構造に対する更新が発生する時、その変更に関連する複数の変数に対する変更を、すべての変更が同時に発生したかのように行わなければならない。他のプロセスは、この更新が実行されている間、これらの共用データにアクセスすることができない。単一の予約では、このような動作を実施するのに不十分である。複数の変数を伴う複雑な更新を必要とする典型的なデータ構造には、待ち行列、優先順位リストおよびデータ・ベースが含まれる。

【0025】本発明で追求する解決策は、下記の要素のすべてを含まなければならない。

1. 更新すべきすべての変数を更新する単一の命令。
2. その単一命令を異なるプロセスがその間にアクセスを許されない状態で更新が実行されるように、実行するための手段（更新の原子性のために必要）。
3. 複数の共用変数を予約する手段。
4. 他のプロセスによって加えられる、予約された変数に対する変更を検出する手段。
5. そのプロセスによって置かれた予約が現在存在することに依拠して、共用変数の原子的更新を実行する手段。

【0026】従来技術には、これらの要因のすべてに対処する解決策がない。クリティカル・セクションは、1つの更新を実行するために複数の命令を必要とするので、信頼性と性能の問題を生じる。様々な単一命令の読取り/変更/書き込み命令は、単一サイクルで非常に特定の更新を実行できるが、より複雑な更新は実行できない。比較交換命令は、1つまたは2つの共用変数を更新する能力を提供するが、変数が連続して保持されることを保証しないので、ABA問題の存在を検出できない。単一予約の使用は、単一変数の更新だけに適しており、複雑な共用データ構造の更新に必要な機能を提供できない。

【0027】したがって、複雑な共用データ構造を原子的に更新できる方法および装置が、この分野で以前から求められていた。このような機構は、上に挙げた要素をすべて必要とする。これを用いると、プログラムが、特定のデータ構造用に存在する待機/解除命令などの複雑な原子的動作の利点をすべて有する、カスタマイズされた読取り/変更/書き込み処理を作成できるようになるは

ずである。

【0028】参考文献

Coffman, E.G., Jr., M.J. Elphick, and A. Shoshani, "System deadlocks," Computing Surveys, Vol. 3, No. 1, pp. 67-78, 1971.

Digital Equipment Corporation, Alpha system reference manual, 1992.

Kane, G., MIPS RISC Architecture, Prentice Hall, 1989.

10 Stone, H.S., High Performance Computer Architecture, Second Edition, Reading MA: Addison-Wesley, 1990.

Sweazy, P. and A. J. Smith, "A class of compatible cache-consistency protocols and their support by the IEEE Futurebus," Proceedings of the 13th Annual International Symposium on Computer Architecture, Tokyo Japan, pp. 414-423, June, 1986.

【0029】

20 【発明が解決しようとする課題】本発明の目的は、予約を使用して、複数のプログラムまたはプロセッサが共用する変数を原子的に更新する、改良された方法を提供することである。

【0030】本発明のもう1つの目的は、複数の予約を使用して、複数のプログラムまたはプロセッサが共用する変数のグループを原子的に更新する、改良された方法および装置を提供することである。

【0031】本発明のもう1つの目的は、シングル・プロセッサ・アルゴリズムの効率的な修正を可能にして、それがマルチプロセッサ環境内またはシングル・プロセッサ上のマルチプログラム環境内で走行できるようにする方法および装置を提供することである。

【0032】本発明のもう1つの目的は、通常でもマルチプロセッサ内に存在するはずのプロトコルとハードウェアをできる限り活用して、本発明によって提供される機能を、最小限の費用と設計労力で既存技術に追加できるようにすることである。

【0033】

【課題を解決するための手段】本発明は、複数の予約を使用して、共用変数のグループを原子的に更新する、効率的で正確な方法である。本発明では、複数の予約レジスタを使用して、複数の共用変数を原子的に更新する。共用変数のグループは、すべて一緒に更新され、あるいはまったく更新されない。

【0034】本発明には、通常は0からN-1までの番号を付けた、任意の数N個の予約レジスタと、複数の共用記憶位置に予約を置くことを含む新規の方法が必要である。共用記憶位置のどれかを現情報で更新する前に、その予約状況を検査する。ある命令によって指定されるすべての位置の予約が、特定の予約状況情報によって有効と判定される場合、修正された共用記憶位置の値がす

(7)

特開平6-222936

11

べて原子的に更新される。1つまたは複数の予約された共用変数の有効性状況記憶が無効状態である場合、どの共用記憶位置でも更新は実行されない。試みられた更新の結果は、条件コードで報告され、その値が更新が行われたか否かを表す。プロセスは、条件コードをテストし、試みが失敗した場合、もう一度その共用変数の原子的更新の実行を試みることができる。

【0035】この方法では、3つの命令、すなわち予約付きロード(LR)命令、条件ストア(Store Contingent, SC)命令および予約時書き込み(WR)命令を使用する。あるプロセスが、1つまたは複数の共用記憶位置の原子的修正を実行する際、そのプロセスは、共用記憶位置を読み取り、その内容を修正し、新しい値を原子的に書き戻すというステップ・シーケンスを実行する。このプロセスは、共用記憶位置のそれぞれをLR命令を用いて読み取ることによって、この原子的更新を開始する。この命令の効果は、更新処理を実行中のプロセッサ内のレジスタに共用記憶位置をコピーすること、およびこれらの変数の予約を同一のプロセッサ内に置くことである。好ましい実施例では、この予約が、LR命令によって指定される予約レジスタに記憶されるアドレスによって表される。LR命令は、同一の予約レジスタ内の特定の状況フィールドの初期設定も行う。

【0036】この時点で、共用記憶域から取り出された共用変数のコピーに対して任意の汎用動作を実行することができる。たとえば、個々の変数を増分、減分またはシフトでき、共用変数を加算または乗算によって組み合わせることができ、共用変数を他の共用変数への間接ポインタとして使用することによって新しい値を得ることができる。これらの動作により、共用変数の1つまたは複数の局所コピーが修正される。その後、SC命令を使用して、修正結果を一時記憶域に記憶する。好ましい実施例では、この一時記憶位置も、共用変数記憶位置に関連する予約レジスタの1フィールドである。また、SC命令は、特定の予約状況フィールドを、共用変数記憶位置に書き戻すべき修正結果が存在することを示すように更新する。マルチプログラム式ユニプロセッサ上であれ、マルチプロセッサ上であれ、プロセスが命令の実行を継続する際には、このようにして複数の共用変数にアクセスし、それを修正することができる。

【0037】ある時点で、これらの1つまたは複数の修正結果を、そのそれぞれの共用変数記憶位置に書き戻さなければならない。これらの更新は、WR命令によって実行される。WR命令は、検査を行って、指定されたレジスタに保持される予約が有効なままであることを確認し、そうである場合には、その変数の局所コピーが修正されていることをその予約状況レジスタが示す共用変数を更新する。共用変数の修正は原子的に行われ、したがって、すべての修正が行われるか、あるいは修正がまっ

12

たく行われぬかのいずれかであり、すべての修正が行われる場合には、他のプロセスがその間に共用記憶位置のどこかに書き込むことはできない。WR命令によって指定されるすべての予約は、そのWR命令の実行の最後に無効状態になり、WR命令は、原子的更新が成功裡に完了したか否かを、条件コードによって報告する。

【0038】

【実施例】図1に、本発明を利用する全体システムを示す。この図は、それを介して制御およびデータを交換することのできる相互接続ネットワーク20を介して互いに接続された複数のプロセッサ21からなる、マルチプロセッサ・システムを示す。相互接続ネットワークは、バスのように簡単なものでも、たとえばStone [1990]の第6章に記載されたものなど、クロスバー・ネットワークや多段相互接続ネットワークのように複雑なものでもよい。相互接続ネットワークには、周知の大域共用記憶域すなわち主記憶域25も接続されている。並列処理環境では、主記憶域25は、1つまたは複数の並列プロセスまたはプロセッサ21によって使用される共用変数を含む、1つまたは複数の共用記憶位置28を有する。共用記憶位置28のコピーが、プロセッサ21上、たとえばキャッシュ24内に存在することができる。

【0039】各プロセッサは、図1の破線21内に示された諸ブロックからなる。これには、プロセッサ・レジスタ22、演算ユニット27、制御ユニット23、キャッシュ24および複数の予約レジスタ26が含まれる。通常、プロセッサ・レジスタ22は、複数のレジスタを有する。本発明者等が知る限り、従来技術の計算機には複数の予約レジスタを有するものはないが、その点を除いて、従来技術では、図1に示したすべての構成要素がシステムに含まれる。1つの予約レジスタを有する計算機の例が、MIPS R4000 [Kane, 1989]である。

【0040】予約レジスタのブロック26は、複数の予約レジスタのバンクを表す。図2に、本発明で使用する好ましい予約レジスタ30を示す。複数のN個の予約レジスタ26のうちの好ましい各予約レジスタ30、ResReg[j]は、ResReg[j].Add32、ResReg[j].S34、ResReg[j].U36、ResReg[j].P38、およびResReg[j].Data39を含む、少なくとも5つのフィールドを備える(各フィールドの添字jは、そのフィールドがどの予約レジスタに含まれるのかを示す)。ResReg[j].Add32フィールドは、LR命令によって予約される共用記憶位置のアドレスを記憶する。このフィールドのフィールド長は、予約を必要とする可能性のある共用記憶位置の最大アドレスのサイズによって決定される。この一時アドレス記憶位置ResReg[j].Add32フィールドは、好ましい実施例では

(8)

特開平6-222936

13

予約レジスタ30内に置かれる。しかし、代替実施例では、この情報を、キャッシュ記憶域内または局所プロセッサ内の緩衝記憶域内に一時的に記憶することもできる。ResReg[j].S34フィールドは、予約レジスタ30が予約を保持しているか否かを示す、通常は1ビットの状況フラグである。言い換えると、このフラグは、そのアドレスが予約レジスタ30内に現れる共用記憶位置28内の変数の予約が有効なままであるのか、それともその予約が遠隔プロセッサまたは遠隔プロセスの処置によって無効化されたかを示す。共用記憶位置28のアドレスが予約レジスタ30のResReg[j].Add32フィールドに記憶される時、共用記憶位置28がレジスタに関連付けられる。

【0041】同様に、ResReg[j].U36フィールドは、通常は1ビットの状況フラグであり、予約レジスタ30に関連する共用記憶位置28を更新することになる修正結果が存在するかどうかを示す。ResReg[j].P38フィールドは、その位置が書き込み特権を有するか否かを示す状況フィールドである。プロセッサは、ある変数を更新する前に、その変数に関する書き込み特権を有する必要がある。プロセッサは、WR命令によって指定される共用変数の原子的更新を開始する前に、そのWR命令によって指定される共用変数のすべてに関する書き込み特権を有する必要がある。これについては、後で図8に関して詳しく説明する。

【0042】ResReg[j].Data39フィールドは、好ましい実施例では、共用記憶位置に書き戻さなければならない修正結果を記憶するのに使用する。好ましい実施例では、修正結果は、予約レジスタ内に一時的に保持されるが、他の実施例では、これらの結果を、キャッシュ記憶域や高速緩衝記憶域など、他の都合のよいどの一時記憶位置に保持することもできる。

【0043】本発明は、複数の予約レジスタを有する新規のコンピュータ装置、およびこの複数の予約レジスタを有効に利用して、マルチプロセッシング環境で共用変数を更新する方法である。この更新は、原子的な1組の修正がすべて実行されるか、あるいはまったく実行されないという意味で、原子的に行われる。さらに、修正の集合を原子的に実行する際に、他のプロセッサは、その間にその組に含まれる予約された位置への書き込みを実行できない。言い換えると、共用変数に対する変更がすべて同時に発生したかのように、それらの変更が他のプロセッサに見えるように更新が行われる。本発明の好ましい実施例は、予約レジスタを用いてWR命令によって予約された共用変数を更新しようとする試みが行われる時に、必ずすべての変更が発生するか、あるいはまったく発生しないように、このような更新を実施する方法を示すものである。

【0044】図3に、本発明の全体的方法を示す。好ましい方法では、下記の3つの命令を使用する。

14

1. 予約付きロード(LR)命令 この命令は、所与の記憶アドレスに関連する指定された予約レジスタに、予約を置く。

2. 条件ストア(SC)命令 この命令は、原子的更新に備えて、共用記憶位置に関連する予約レジスタの1フィールドに修正結果の値を置く。

3. 予約時書き込み(WR)命令 この命令は、そのアドレスが有効な内容と共に予約レジスタに保持され、予約を置いた後にSC命令によって更新された共用記憶位置の原子的更新を実行する。原子的更新は、すべての予約が有効であり、すべての予約に関して書き込み特権が成功裡に取得されている場合に行われる。これらの条件が満足されない場合、予約が削除され、変数は変更されない。

【0045】方法300は、ブロック302から開始し、プロセスが、LR命令を用いて、最初の共用記憶位置にアクセスする。予約を置く方法の正確な詳細については、後でより詳しく説明する。

【0046】ブロック304で、局所プロセッサ内の予約レジスタの数までの、原子的更新が依存する数の共用変数に対して、追加のLR命令による予約を置く。

【0047】ブロック306で、計算機レジスタ内で共用変数の更新済み値を生成する計算を実行する。これらの変数は、ブロック308で、SC命令によって予約レジスタにセーブされる。予約された変数のすべてが更新されることも、予約された変数の一部だけが更新されることもある。更新された変数のすべてが予約されていない。さらに、原子的更新の整合性のために、その更新で共用変数の最新の値を使用する必要がある場合には、他の共用変数が更新されない場合でも、そうした他の共用変数が予約されていない。LR命令は、更新が依存する各共用変数ごとに必要であり、また原子的動作中に更新される変数に対しても必要である。たとえば、これらの共用変数には、ある処理手順の入出力が含まれる可能性がある。

【0048】ブロック310で、単一のWR命令の実行によって原子的更新が実行される。この例では、遠隔プロセッサが予約された共用変数に書き込んでおらず、したがってブロック310に達した時にすべての予約が有効であるので、この更新が実行される。

【0049】ブロック312で、プロセスが、WR命令の返す条件コードをテストし、更新が行われたことを知る。この場合には、流れ図の成功側の出口に進む。

【0050】図4は、図3に非常に類似しているが、この例では、遠隔プロセッサまたは遠隔プロセスが、ある予約された共用変数を修正し(それに書き込み)、予約を無効化する点が異なる。図4の処理400は、ブロック402から始まり、すべての予約が置かれる。ブロック404で、変数の更新された値を計算する。ブロック406で、遠隔プロセッサまたは遠隔プロセスによって

(9)

特開平6-222936

15

共用変数が修正され(書き込まれ)、局所プロセッサ内のその変数の予約が無効化される。ブロック408で、プロセッサが、SC命令によって、更新された値をすべて予約レジスタに記憶し、ブロック410で、WR命令の実行を試みる。ブロック406で無効な予約が行われたので、何も更新されない。ブロック412で、プロセスがWR命令の条件コードをテストし、失敗の出口に進んで、ブロック402でこの原子的更新を繰り返す。代替実施例では、遠隔プロセッサまたは遠隔プロセスが共用変数へのアクセスを実行することによって共用変数の予約を無効化するが、このアクセスは、この場合のように共用変数を修正するものであっても、そうでなくてもよい。

【0051】図5に、LR命令の好ましい実施例を示す。好ましい実施例では、LR命令により、ResReg[j]で表される予約レジスタ26、記憶位置X、およびプロセッサ・レジスタ22 Reg[i]を指定する。

【0052】図5で、Xの予約をResReg[j]に置くために、このプロセスは、キャッシュ・アクセス(ブロック40)から開始、キャッシュ内にXのコピーがあるかどうかをテストする。この実施例では、まず、共用変数のコピーが局所キャッシュ記憶域内にあるかどうかを調べる。その項目がキャッシュ内にない場合、ブロック45の条件テスト「ヒットか」の「No」出口に沿って、プロセッサは、図1の相互接続ネットワーク20にXの値を求める要求を置く。プロセッサは、図5のブロック43でXの値を待ち、それが到着した時、ブロック44で、ResReg[j].AddにXのアドレスに対する予約を置く。Xの予約を置く動作では、そのアドレスをResReg[j].Addに記憶し、Sビット(状況ビット)を、レジスタが予約を保持していることを示す1にセットし、Uビット(更新ビット)を、その値が更新されていないことを示す0にセットし、Pビット(特権ビット)をプロセッサがXに関して保持している現在の書き込み特権にセットすることが必要である。プロセッサが書き込み特権を保持している場合、Pビットは1にセットされる。そうでない場合、Pビットは0にセットされる。書き込み特権は、通常はXのキャッシュ・エントリに記憶され、ブロック40でのキャッシュ・アクセスの後に局所プロセッサが利用可能になる。Xがキャッシュ内に存在しない場合、プロセッサは、ブロック43で待った後にXを受け取る時、通常は書き込み特権を受け取らない。ただし、Xが他のキャッシュに保持されていないことを検出できるシステムの場合には、この時点で書き込み特権を与えることができる。

【0053】キャッシュの一貫性を維持するため、どの位置Xについても、任意の所与の時点に、せいぜい1つのプロセッサがXに対する書き込み特権を有するようにシステムを構成しなければならない。従来技術の慣行で

16

は、Sweazy and Smith [1986]に記載の書き込み特権によってキャッシュ一貫性を維持するシステムを作成する。

【0054】図5のブロック44で、本発明の好ましい実施例で使用するUビット(更新用)とPビット(書き込み特権用)を新規に追加する。これらの予約パラメータを用いると、マルチプロセッシング環境で複数の共用変数を原子的に更新できるようになる。というのは、これらのパラメータは、書き込み特権を有しておらず、原子的更新を開始する前にそれを得なければならない予約を示し、また、実際に変更され、したがって原子的更新中に共用記憶域へ再書き込みする必要のある予約された変数を示すからである。

【0055】図6に、SC命令の動作を示す。この命令は、すべての修正された変数の原子的更新に備えて、共用変数の新たに更新された値をResReg[j].Dataに置く。この命令は、ResReg[j].U36ビットを1にセットすることによって、予約を更新済みとしてマークする。この動作は、ブロック59で行われる。修正結果は、最終的には、同じ予約レジスタのResReg[j].Add32フィールドにそのアドレスがセーブされている、共用記憶位置に置かれる。修正結果を予約レジスタから共用記憶域にコピーする動作が、成功裡の原子的更新中に行われる。原子的更新に失敗した場合、修正されたデータは、WR命令が原子的更新を行う試み中に失敗した時に破棄される。

【0056】図7に、WR命令の全体的構造を示す。後の議論で、この図に含まれる特定のブロックについて詳細に説明する。WR命令は、後で図8に関して説明する1組の予約レジスタを指定する。これらの予約がすべて、WR命令の実行の時点で有効であり、かつ、1つまたは複数の予約が、予約された共用変数中にセーブしなければならない新たに修正された値を有する場合、WR命令は、すべての変数の値を原子的に書き込もうとするが、原子的更新を行えない場合には、まったく書き込まない。

【0057】原子的更新を実行するためには、すべての更新された変数の書き込みが、すべての選択された予約された変数の現在の値に基づいている必要がある。選択された予約に現在値が含まれることを保証するには、選択された予約された変数の書き込み特権を得ることが必要である。これらの変数のどれかが、予約が置かれて以降に値を変更された場合、または、すべての予約に関して書き込み特権が得られる前にいずれかの予約の値が変化した場合には、WR命令は、失敗の出口に進み、変数は変更されない。

【0058】命令が、すべての選択された予約された変数に関する書き込み特権の取得に成功し、すべての選択された予約が、この時点で有効なままである場合には、WR命令は、コミット段階に入り、その間に、更新された

(10)

特開平6-222936

17

変数がすべて、同時に更新されたかのように更新される。すなわち、いずれかのプロセッサに、この時点の後に共用変数のうちの1つの更新済みの値が見える場合、そのプロセッサに、他のすべての変数の未更新の値が見えてはならない。原子的更新と予約の保持を制御する方式では、予約レジスタを有さないシステム内に存在しなければならないキャッシュー貫性プロトコル・メッセージを利用する。このようなシステムは、Sweazy and Smith [1986] に記載されている。したがって、システム内に既に存在しているはずのものに比較的小さな追加の複雑さを導入することによって、キャッシュー貫性を有するマルチプロセッサ・システム内で複数の予約を実施することができる。

【0059】図7のブロック60で、WRフラグが1にセットされ、この命令が開始されたことを示す。この期間中にプロセッサが受け取るキャッシュー貫性メッセージは、このフラグが0状態に戻るまで延期される。このフラグのテストについては、後で説明する。好ましい実施例では、WRフラグは、図1の制御ユニット23内に置かれる。

【0060】ブロック61の予約テスト論理回路は、選択されたすべての予約のSビットが、すべての予約が有効なままであることを示す1であることを検証する。遠隔プロセッサが、予約された変数に新しい値を書き込む場合、その動作が、キャッシュー貫性メッセージを介してこのプロセッサに通信され、Sビットを0に変更することによって予約を無効化する。このことについては、後で詳細に説明する。ブロック62のテストですべての予約が有効である場合、この命令は、もう一度有効性を再テストする前に、もう1つの予約された共用変数の特権を獲得しようと試みる。特権は、ブロック63で候補の予約を見つけることによって獲得される。このことについては、後で図9に関して説明する。その後、ブロック64で、候補が残っているかどうかをテストする。そうである場合、ブロック68で特権を要求し、この命令は、ブロック69で応答を待つ。応答が到着した時、ブロック70で、選択されたレジスタのPビットを更新し、ブロック61の選択された予約の再テストから始まるループを繰り返す。このループを回っている間に、遠隔プロセッサからのメッセージが到着し、選択された予約が無効化される可能性がある。したがって、選択された予約を、ブロック61で繰り返しテストする。すべての特権を獲得し終えた時、この命令は、コミット処理が進行中であることを示すフラグをセットすることによって、ブロック65から始まるコミット段階に入り、ブロック66で、コミット処理を実行する。

【0061】ブロック67には、コミットの成功に必要なハウスキーピングが含まれ、ブロック71には、この命令の非成功の実行に必要な、類似のハウスキーピング動作が含まれる。その詳細については、後で説明する。

18

【0062】図8に、予約テスト（図7のブロック61を参照）に必要なハードウェア論理回路を示す。命令レジスタ75内のビットは、WR命令に参加する予約を指定する。この例では、ビット位置0および3の1が、ResReg[0]とResReg[3]の参加を示す。

【0063】図8では、予約レジスタ30のSビットが示されている。この例では、命令によって指定された2つのレジスタ、ResReg[0]とResReg[3]が、そのSビットが1にセットされ、その予約が有効なままであることを示している。ブロック77で、選択ビットの補数とSビットの真値の論理和を形成する。その出力は、レジスタが選択されていない場合、またはSビットが1の場合に、論理1になる。すべてのORゲートがその出力上に1を有する場合、ANDゲート（ブロック78）は1の出力を発生し、そうでない場合は、0を発生する。この出力は、図7のブロック62でテストされる。

【0064】図9に、書き込み特権要求の候補を選択するための論理回路（図7のブロック63）を示す。この図で新規な点は、要求が、昇順アドレスの順序に並べられることである。その意図は、書き込み特権を求めて衝突する入力要求を、可能な時、延期させることである。しかし、特別な処置を講じない限り、外部から発行された要求の完了を待っている間、入力要求を延期させる処置は、複数のプロセッサが互いに無限に待ちそれ以降の進行がまったく不可能になる、デッドロック状況をもたらす可能性がある。従来技術では、すべてのプロセッサによる要求を固定順に置くと、デッドロックが発生しないことが知られている【参考：Coffman, et al. 1971】。本発明では、図9に示すように、この原理に従って、書き込み特権のない最小の選択されたアドレスを選択する。

【0065】WR命令は、命令レジスタ75内に常駐し、1つまたは複数の予約レジスタを指定する。図9の予約レジスタ30は、予約のアドレスと、それが書き込み特権を有するかどうかを示すPビットを含む。ANDゲート80は、対応する予約レジスタが選択され、書き込み特権を有しない場合に、1のビットを発生する。図9のソート機構81は、そのアドレス出力に候補のアドレスを出力し、そのNoneValid（有効なし）出力に2進信号を出力する。候補がない場合、NoneValidビットは論理1を有し、そうでない場合には論理0を有する。NoneValidビットは、図7のブロック64でテストされる。

【0066】図7のWR命令の成功終了（ブロック67）と失敗終了（ブロック71）について、図10で詳細に説明する。結果を示す条件コードがセットされ、この条件コードは、失敗の場合は0（ブロック85）、成功の場合は1になる（ブロック82）。どちらの場合でも、終了の動作で、ブロック83でWRフラグがリセット

(11)

特開平6-222936

19

トされ、要求の通常処理が行えることを示す。ブロック84で、WR命令の実行中延期されていた入力要求の処理を実行する。これについては、後で詳細に説明する。

【0067】WR命令の心臓部は、コミット段階（図7のブロック66）である。これについて、図11で詳細に説明する。ブロック90で、選択された1組の更新済み予約中から、ある更新済み予約を選択する。ブロック91で、候補が残っているかどうかを判定する。残っている場合、ブロック92でその局所キャッシュ内の候補を更新し、Sビットを0にセットして、その予約が削除されたことを示す。ブロック90に戻って、もう一度候補を選択するが、Sビットがリセットされているので、先程処理された候補は選択されない。追加の更新を実行する必要がなくなった時、ブロック94で、他のすべてのSビットをリセットし、ブロック95で、コミット・フラグをリセットして、コミット段階が終了したことを示す。コミット段階中は入力要求の特殊処理が呼び出されるが、コミット・フラグをリセットすると、入力要求の通常処理が再開できるようになる。

【0068】図12に、プロセッサ21の制御ユニット23内にある、入力メッセージ受取り機能の論理機能を示す。少なくとも3種類のメッセージを受け取ることができる。1つは、書き込み特権を求める要求に対する応答である。2番目は、書き込み特権を求める要求である。3番目は、書き込み特権を有するプロセッサがキャッシュ・エントリを変更したので、キャッシュ・エントリの無効化を求める要求である。他のメッセージを受け取ることもあるが、これらは予約機構に関係ない。

【0069】ブロック100、102、104で、受取り機能がテストを行って、入力メッセージが上で述べた3タイプのうちの1つであるかどうかを調べる。そうである場合、受取り機能は、ブロック101、103、105に示すそれぞれの処理を実行する。未処理の書き込み特権要求に対する応答を受け取ると（ブロック100）、受取り機能は、ブロック101で、WR命令に、図7のブロック70での処理を再開するよう通知する。書き込み特権を求める要求を受け取った時（ブロック102）、局所プロセッサがそのような特権を有する場合には、ある条件の下ではその書き込み特権を認めるが、他の条件の下ではこれを延期し、後で認める。これは、図12のブロック103で実行される。これをどのように実行するかについては、後に説明する。

【0070】無効化要求を受け取った場合（ブロック104）、その無効化は、後で説明するように即座に実行される。その処置は、ブロック105によって表される。

【0071】図13に、デッドロックを発生しない形で要求を延期するための論理を示す。その目的は、昇順アドレスの順序で書き込み特権を要求することである。プロセッサは、それが未処理の書き込み特権要求を有する最高

20

のアドレスより低位のアドレスに対する書き込み特権の要求を延期することができる。プロセッサは、その現書き込み特権要求のアドレスより高位のアドレスに対する書き込み特権を保持することができる。その場合、デッドロックを回避するために、プロセッサは、そのようなアドレスに対する要求が到着した場合に、それらに対する書き込み特権を放棄しなければならない。

【0072】図13のブロック110で、メッセージ受取り機能は、入力特権要求が、Sビットが1であって、予約が有効であることを示す、予約レジスタに保持されたアドレスに対するものであることを検出する。WR命令がコミット段階にある場合、その要求を、この段階が終了するまで延期することができる。ブロック111でこれをテストし、ブロック112で要求を延期する。コミット段階でない場合、ブロック113で、WRフラグをテストすることによって、WR命令が実行中であるかどうかを調べる。そうである場合、入力要求が特権を求める最後の出力要求のアドレス以下のアドレスに対するものであるならば、入力要求を延期することができる。ブロック114でこれをテストする。他のどの場合でも、その特権要求を認めなければならない。これは、ブロック115で無効化処理を行い、ブロック116で要求された特権を認めることによって行われる。これらの処理については、後で詳細に説明する。

【0073】図14に、無効化処理（図13のブロック115）を示す。ブロック120で、一致する予約レジスタのSビットを0にセットし、これによって、現在のまたは後続のWR命令が成功しなくなる。無効化は、変数の値が変化した、またはごく近い将来に変化し、共用変数の予約値が、もはや現在値とは見なせなくなることを意味する。ブロック121で、プロセッサが、キャッシュから変数を読み取る。ブロック122で、その変数がキャッシュ内にあると判定される場合、ブロック123で、そのキャッシュ・エントリを無効化する。

【0074】図15に、特権認可の既知の処理を示す。ブロック125で、プロセッサが特権の有無についてテストする。これは、キャッシュから読み取られたエントリの内容に対して実行される。ブロック126で、特権が、キャッシュから読み取られたデータと共に要求元に転送される。キャッシュが、それに関連する書き込み特権を有するので、これはデータの現在値であることがわかっている。

【0075】図16に、図10のブロック84で、WR処理がどのように要求を延期するかを示す。図16のブロック130で、最も古い延期された要求を選択することによって、先入れ先出し順に要求を選択する。ブロック131で、延期された要求があるかどうかを調べ、なければ終了する。ブロック132で、延期要求待ち行列からその要求を削除し、ブロック133で、要求されたアドレスを求めてキャッシュにアクセスする。ブロック

(12)

特開平6-222936

21

134で、そのアドレスがキャッシュ内にあるかどうかを調べる。というのは、キャッシュ・エントリが書き込み特権（所有権）を含む可能性があるからである。その項目がキャッシュ内にない場合、処理はブロック130に戻って、次の要求を取得する。

【0076】しばらくの間、項目がキャッシュにあると仮定する。ブロック135で、そのキャッシュ・エントリを無効化する。というのは、書き込み特権が転送され、新所有者が変数の値を変更することになるからである。ブロック136で、キャッシュから読み取ったエントリが書き込み特権を有するかどうかを調べる。そうでない場合、処理はブロック130に戻って、別の延期された要求を取得する。

【0077】現プロセッサが所有権を有する場合、ブロック137で、現プロセッサは特権を認める。同一のエントリに対する延期された要求が他にも存在する可能性があり、新所有者が、これらの要求を既に受け取り、拒絶している可能性がある。したがって、ブロック138で、プロセッサは、同一の項目に対する延期された要求が他にも存在するかどうかを調べる。そのような要求のそれぞれについて、プロセッサは、ブロック139で延期要求待ち行列からその要求を削除し、ブロック140で要求元に否定応答（NAK）を送って、所有権が転送されたのでその要求を繰り返すよう要求元に伝える。

【0078】書き込み特権を求める要求を、図9のブロック81に示すように順序付ける必要はない。要求が順序付けられない場合、図13のブロック113でのWR命令の非コミット段階の処理中に書き込み特権を求める入力要求を延期することはできない。しかし、図13のブロック111に示すように、コミット段階中は、これらの要求を延期することができ、延期しなければならない。

【0079】更新処理の原子性は、本発明により、図12ないし図15に記載のメッセージの処理と相互作用する、図11のコミット論理機構という手段によって保証される。コミット段階が開始すると、無効化動作によって、または遠隔要求に書き込み特権を認めることによって、予約が失われることはない。予約された変数の無効化は、コミット段階中には発生し得ない。というのは、このようなすべての変数に対する書き込み特権が獲得されており、したがって、他のプロセッサがこれらを変更できないからである。選択され予約済み変数に対する書き込み特権を求める要求をコミット段階中に受け取る場合、その要求は、延期され、後で認められる。コミット段階中は、割込みおよびトラップを延期して、開始したコミット段階が完了まで実行できるようにしなければならない。あるプロセッサ内の割込みによって、同一の予約レジスタを使用する別のプロセスにコンテキストが変更される時は、予約を無効化しなければならない。

【0080】本発明を使用して、ユニプロセッサ内で動作するように設計されたアルゴリズムを、マルチプロセ

22

ッサ環境内で動作するように効果的に修正することができる。図17を参照のこと。これを行うには、まずブロック170で、ユニプロセッサ・プログラム内の書き込み命令および読取り命令のうち、マルチプロセッサ環境内の複数のプロセスによる修正の対象となり得る変数、あるいはそれらの変数の更新が依存する変数にアクセスする命令を識別する。ブロック172で、それらの読取り命令をLR命令に変更しそれらの書き込み命令をSC命令に変更する。ブロック174で、最後のSC命令の直後に、LR命令が使用する予約レジスタを指定するWR命令を挿入する。ブロック176で、このWR命令の直後に、最初のWR命令に戻る条件付き分岐を挿入し、WR命令が共用変数を更新しなかった場合に分岐が行われるように分岐論理を構成する。その後、本明細書に記載のマルチプロセッサ環境を構築する。こうすることによって、マルチプロセッシング環境でこのアルゴリズムのマルチプロセッサ版として走行するユニプロセッサ・プログラム用のカスタマイズされた原子的動作が作成される。このマルチプロセッサ・アルゴリズムは、識別された共用変数の集合を修正し、その集合に含まれる変数が、更新の前に別のプロセスによって修正されていないことを保証する。

【0081】これは、すべての種類のユニプロセッサ・アルゴリズムに対する一般的な解決策である。

【0082】この新規開示の知識を有する当業者には、本発明の教示を実行する方法が多数存在することは明白であろう。それらの代替実施例も、本発明者等によって企図されている。たとえば、好ましい実施例では、図12に示したタスクや、コミット・フラグなどの他の制御機能を実行するようなメッセージ処理ハードウェアが、各並列プロセッサ21の制御ユニット23内に置かれる。別法として、このハードウェアを、プロセッサ21を相互接続ネットワーク20に接続するプロセッサ21内のハードウェア・インターフェース内に置くこともできる。また、これらの機能を他の位置に置くことも可能である。さらに、好ましい実施例では、ResReg[j]・S34を使用するが、このResReg[j]・S34を省略することもできる。その場合は、WR命令を実行するたびに、すべての予約レジスタ30が変更されることになる。さらに、共用記憶位置28またはそれらの位置にある値のコピーが、主記憶域25以外の位置に存在できる。この代替位置は、1つまたは複数のプロセッサ21上に常駐できる待ち行列または緩衝記憶域を含むことができる。さらに、図4のブロック406では、予約された位置を遠隔プロセッサが修正する場合に予約が無効化されるが、遠隔プロセッサがその位置からLR命令を実行する場合、または遠隔プロセッサがその位置を修正する場合に、予約を取り消す必要がある代替方法も可能である。

【図面の簡単な説明】

(13)

特開平6-222936

23

24

【図1】複数の予約レジスタを有する、本発明のコンピュータ装置のブロック図である。

【図2】好ましい実施例の予約レジスタ装置を示す図である。

【図3】複数の共用記憶位置を修正済みデータで原子的に更新するのに成功した場合の、本発明の全体的な方法の流れ図である。

【図4】複数の共用記憶位置を修正済みデータで原子的に更新する試みに失敗した場合の、本発明の全体的な方法の流れ図である。

【図5】好ましい実施例で、LR命令を実行するステップを示す流れ図である。

【図6】SC命令を実行するステップを示す流れ図である。

【図7】WR命令を実行するステップを示す流れ図である。

【図8】複数の共用記憶位置の予約状況をテストする論理装置を示すブロック図である。

【図9】書き込み特権を有しない予約レジスタのグループ内から、最小の選択されたアドレスを見つける装置を示すブロック図である。

【図10】特定の予約レジスタ・フィールドが、WR命令によってどのように更新されるのかを示す流れ図である。

【図11】他の特定の予約レジスタ・フィールドが、WR命令によってどのように更新されるのかを示す流れ図である。

【図12】予約プロトコルを実施するためのネットワーク・メッセージ受取り機能の動作を示す流れ図である。

【図13】特権要求が、ネットワーク・メッセージ受取り機能によってどのように処理されるのかを示す流れ図である。

* 【図14】プロセッサが、キャッシュ無効化のためにどのように予約をキャッシュ・プロトコルに組み込むのかを示す流れ図である。

【図15】ネットワーク・メッセージ受取り機能が、どのように要求元に書き込み特権を与えるのかを示す流れ図である。

【図16】延期されたメッセージが、WR命令の完了後にどのように処理されるのかを示すブロック図である。

【図17】シングル・プロセス・アルゴリズムを、シングル・プロセッサ上またはマルチプロセッサ上のマルチプロセッシング環境内で走行するアルゴリズムに変換する方法を示す流れ図である。

【符号の説明】

20 相互接続ネットワーク

21 プロセッサ

22 プロセッサ・レジスタ

23 制御ユニット

24 キャッシュ

25 主記憶域

26 予約レジスタ

27 演算ユニット

28 共用記憶位置

30 予約レジスタ

32 ResReg[j].Add

34 ResReg[j].S

36 ResReg[j].U

38 ResReg[j].P

39 ResReg[j].Data

75 命令レジスタ

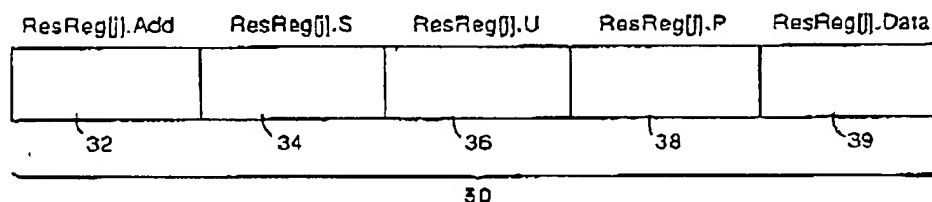
78 ANDゲート

80 ANDゲート

* 81 ソート機構

【図2】

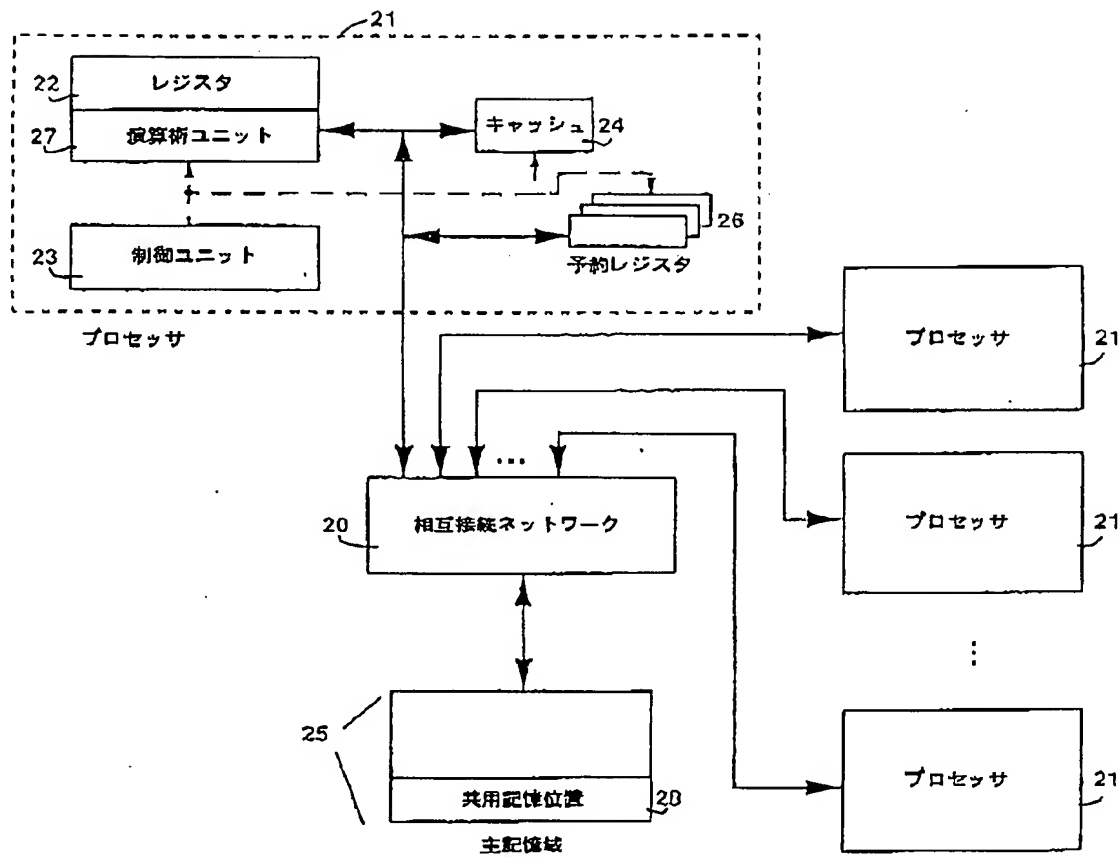
予約レジスタの構造



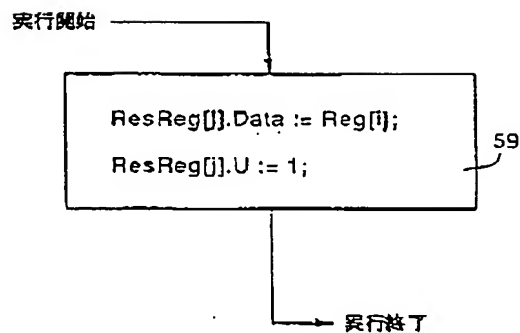
(14)

特開平6-222936

【図1】

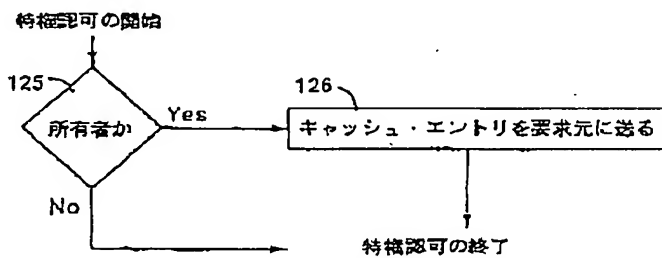


【図6】



【図15】

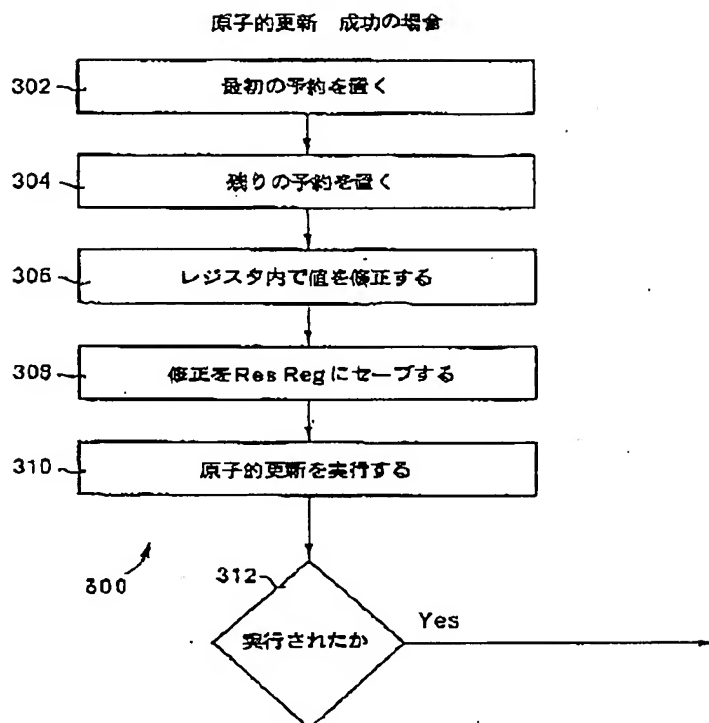
ネットワーク・メッセージ受取り機能、特権認可



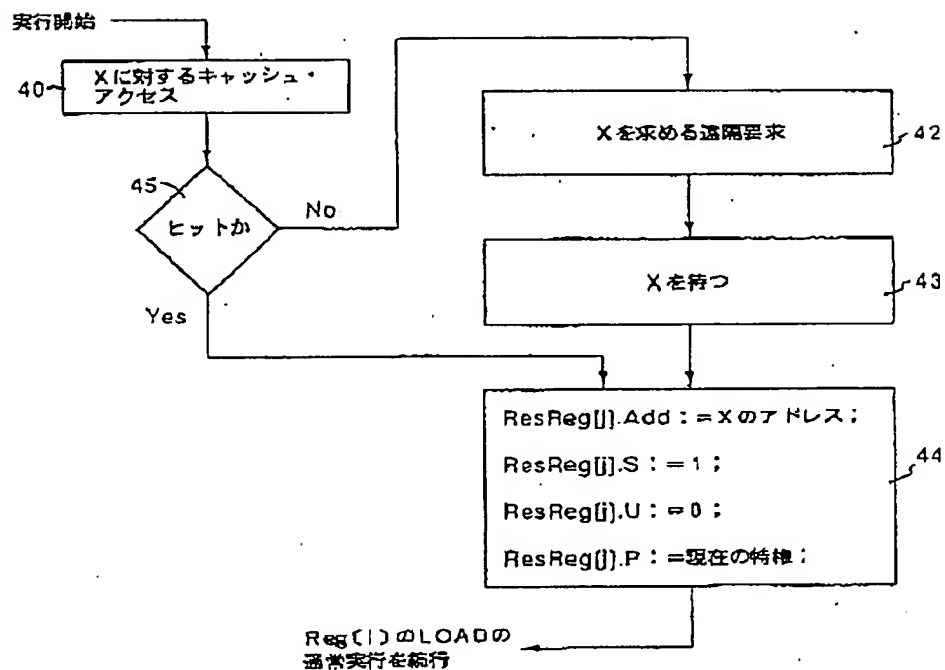
(15)

特開平6-222936

【図3】



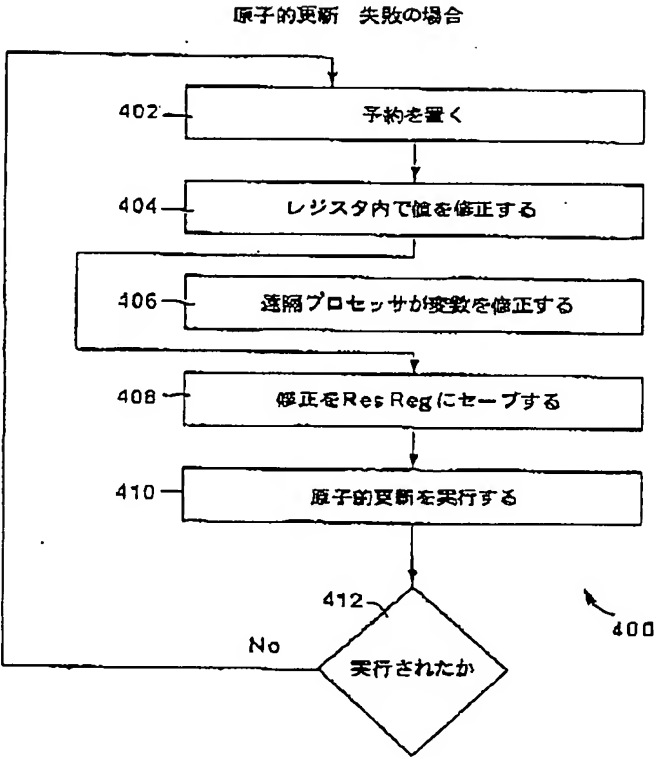
【図5】



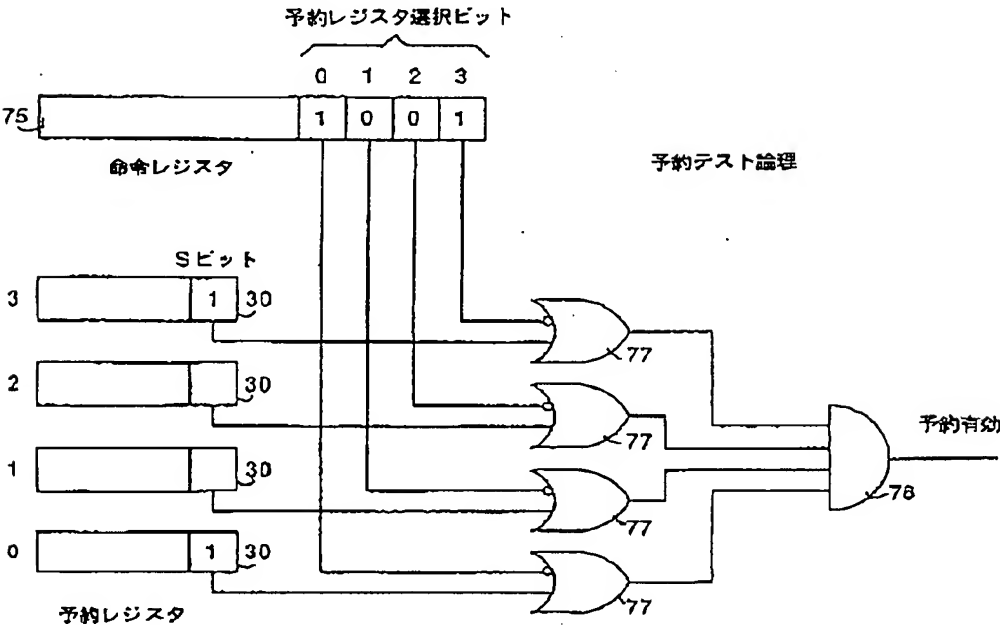
(16)

特開平6-222936

【図4】



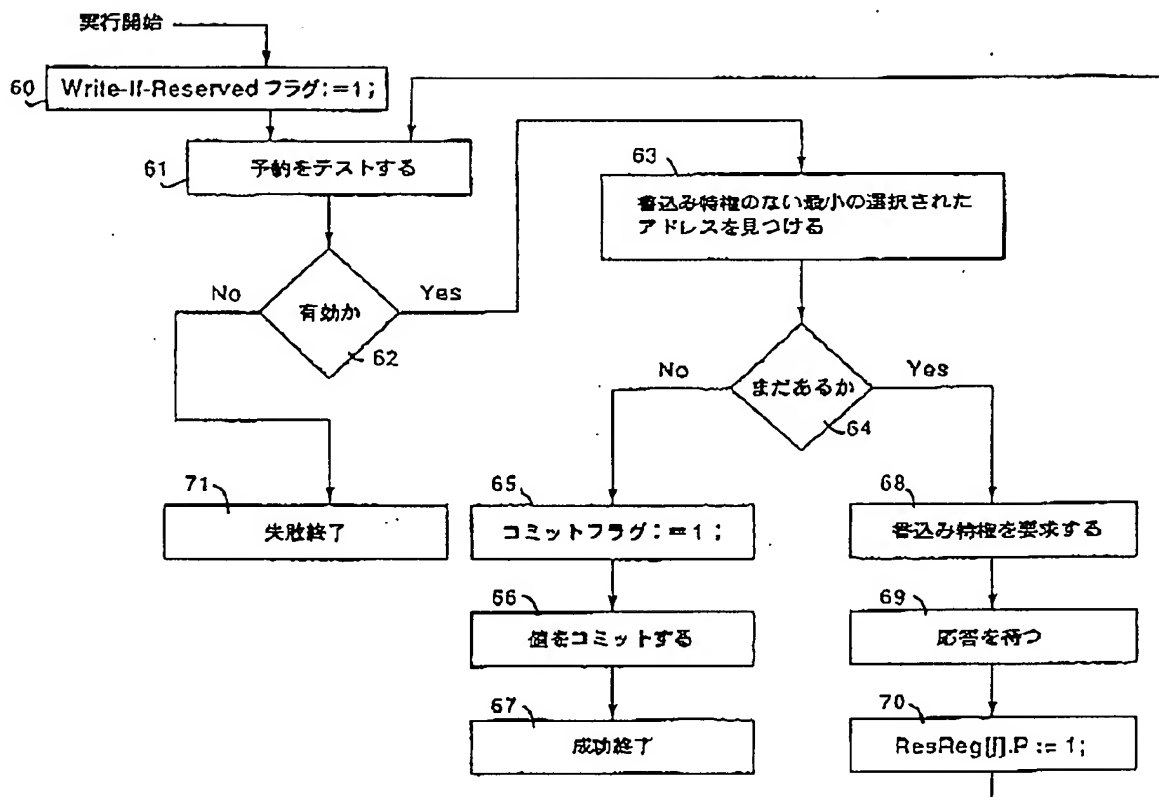
【図8】



(17)

特開平6-222936

【図7】

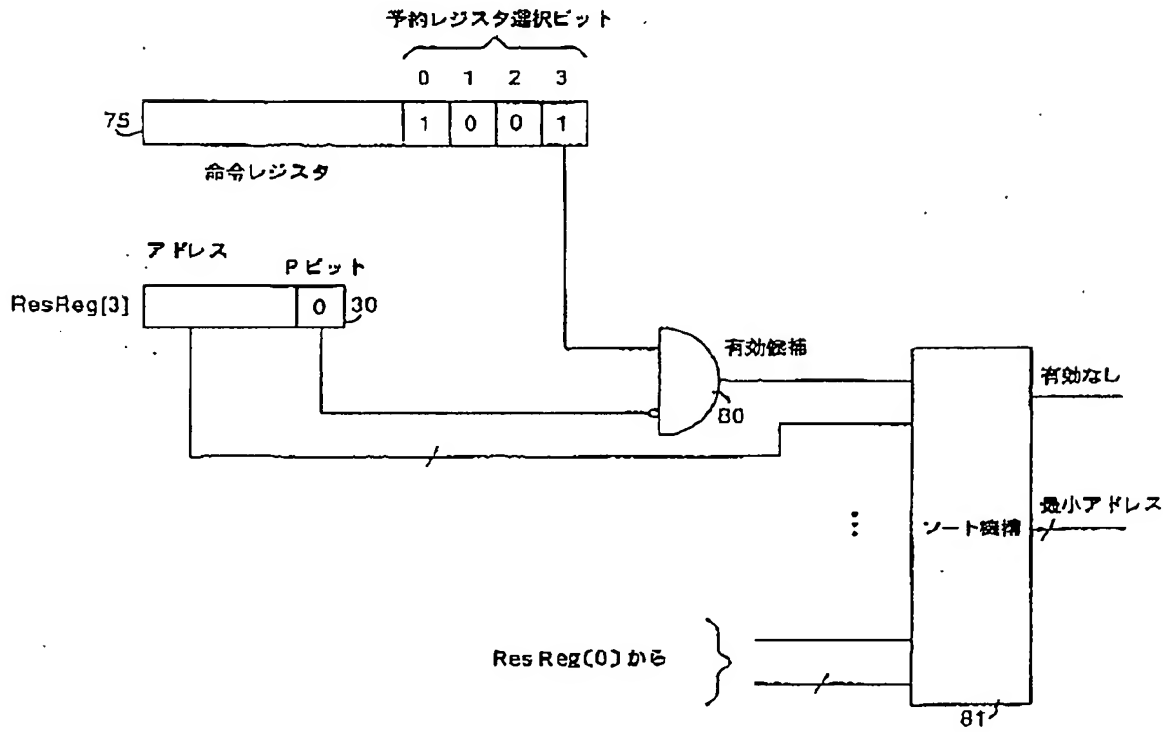


(18)

特開平6-222936

【図9】

書き込み特権のない最小の選択されたアドレスを見つける論理

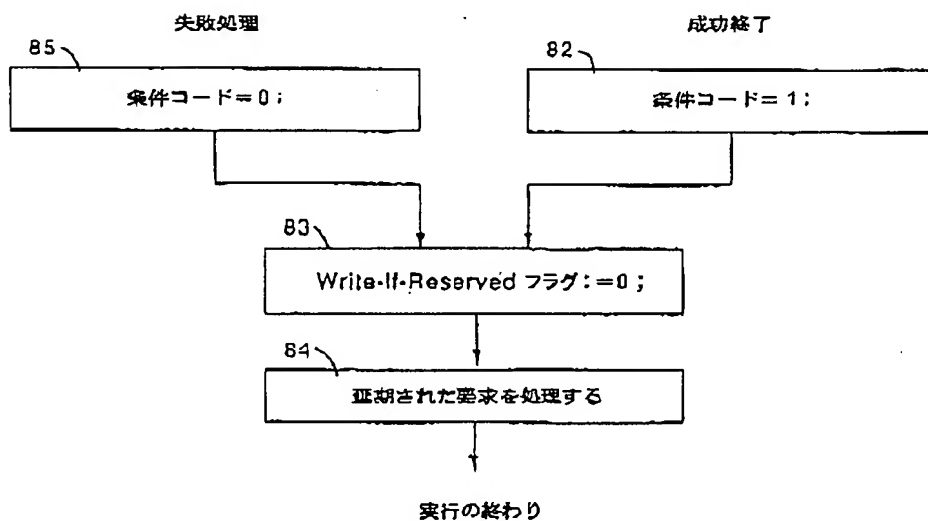


(19)

特開平6-222936

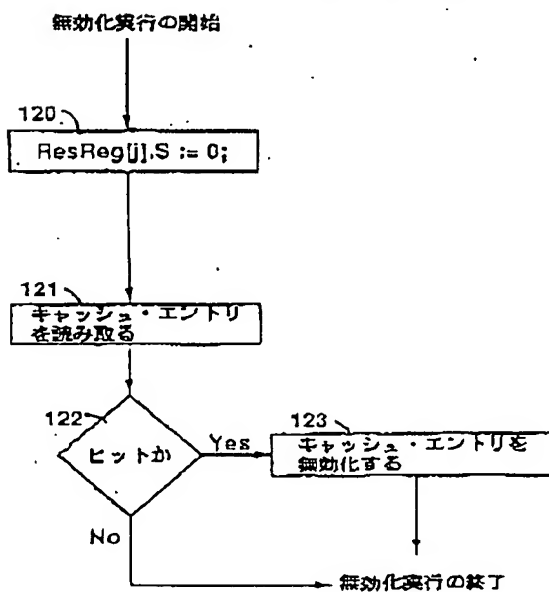
【図10】

予約時着込み処理の終了



【図14】

ネットワーク・メッセージ受取り機能、無効化実行

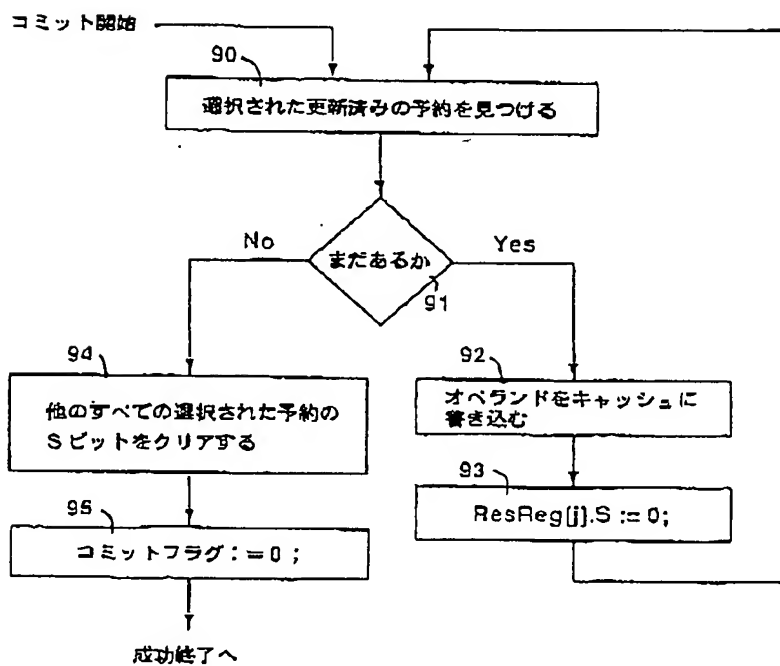


(20)

特開平6-222936

【図11】

予約時書き込み処理、値のコミット

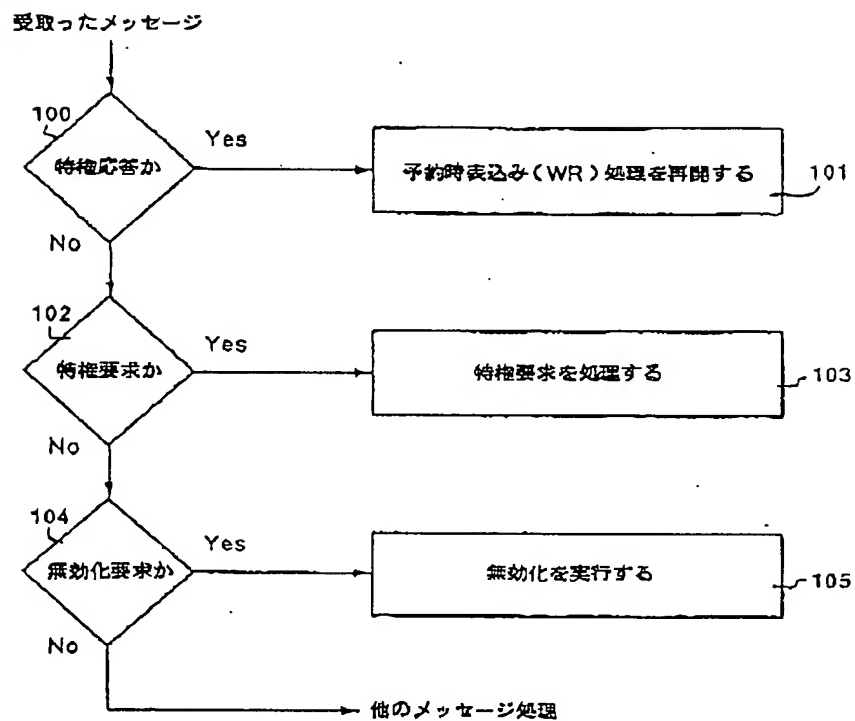


(21)

特開平6-222936

【図12】

ネットワーク・メッセージ受取り機能

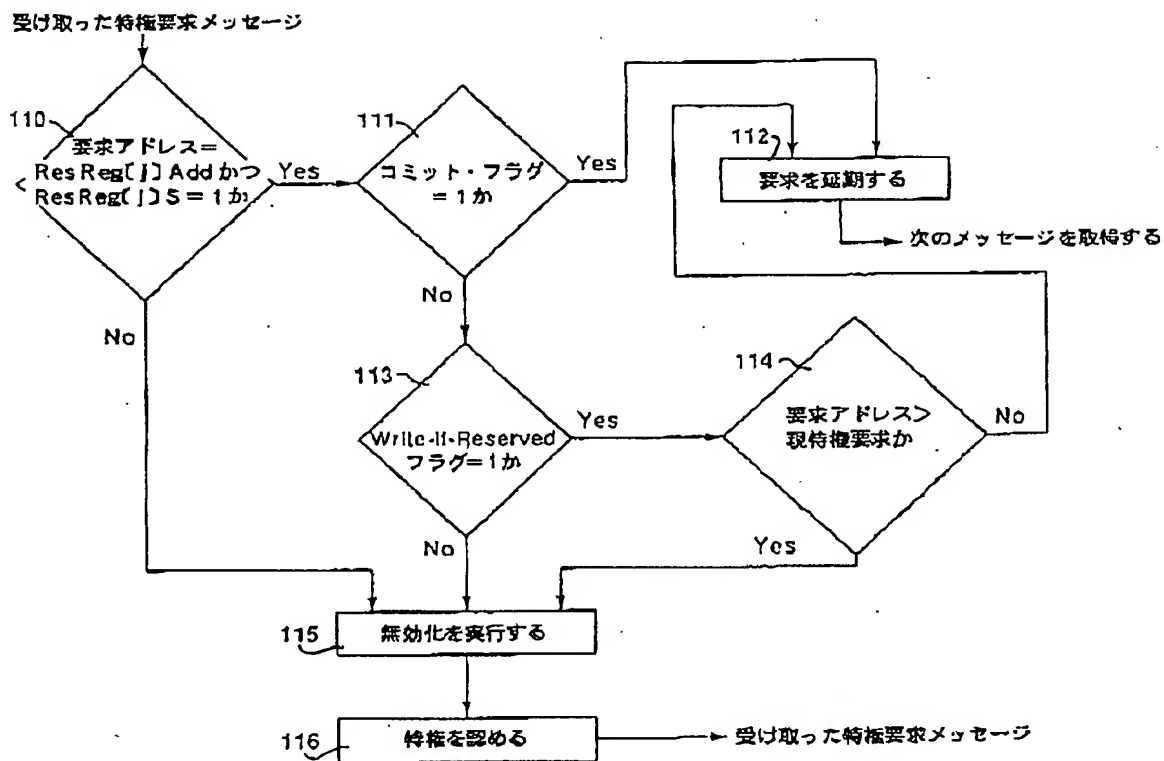


(22)

特開平6-222936

【図13】

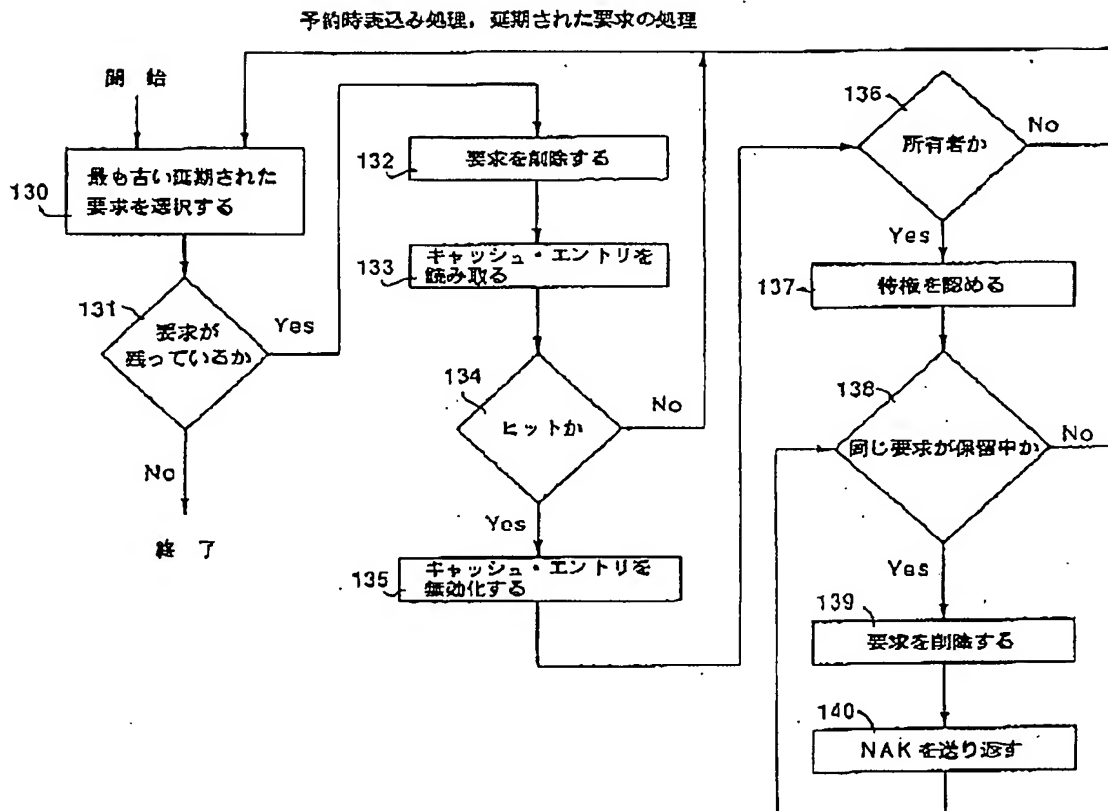
ネットワーク・メッセージ受取り機能、特権要求処理



(23)

特開平6-222936

【図16】

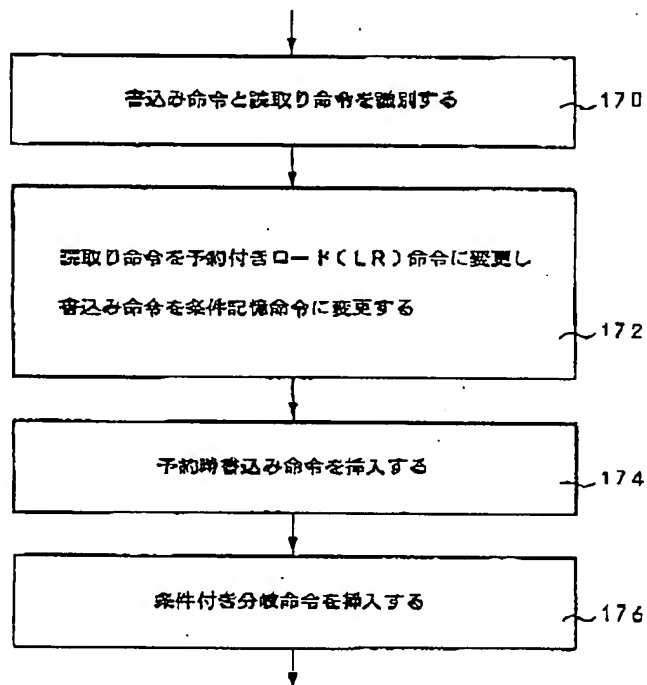


(24)

特開平6-222936

【図17】

順次アルゴリズムをマルチプロセッサ・アルゴリズムに変換する方法



フロントページの続き

(72)発明者 ジャニス・マーフィー・ストーン
アメリカ合衆国 ニューヨーク州チャパク
ァ クロス・リッジ・ロード 50

Reference Number: SCEI02070

Dispatch Number: 034347

Dispatch Date: January 30, 2007

Notification of Reason(s) for Refusal

Patent Application No.	Patent Application No.2004-349195
Drafting Date	January 22, 2007
Examiner of JPO	Masaya TONOKAWA 9646 5B00
Representative / Applicant	Sakaki MORISHITA
Applied Provision	Patent Law Section 29(2)

This application should be refused for the reason(s) mentioned below. If the applicant has any argument against the reason(s), such argument should be submitted within three months from the date on which this notification was dispatched.

Reasons

The inventions in the claims mentioned below of the subject application should not be patented under Patent Law Section 29(2) since they could easily have been made by persons who have common knowledge in the technical field to which the inventions pertain prior to the filing of the subject application, on the basis of the inventions described in the publications listed below which were distributed in Japan or foreign countries or the inventions available to the public via electronic communication lines prior to the filling of the subject application.

Note (See the list of cited documents, etc. below.)

-Claims 1-29

-Cited documents 1-2

-Remarks

Cited document 1 describes the computer including common memory type multiprocessor configuration with a plurality of processor with a cache memory, which changes its mode to the low-power consumption mode when it detects the execution of spinwait command, or executes other process in order to perform the barrier synchronization ("commencing the other processing task"), and

further describes the feature, whereby the change of value is posted to the processor by detecting the change of state at the value change detecting unit when the cache state is updated from "Share" (S) to "Invalidation" (I).

Cited document 2 discloses the technology, which makes use of the load with reservation command and write command at reservation ("notification of reservation cancellation") to ensure that the change of status is detected safely at the value change detecting unit. Therefore the invention according to claim 1 of the subject application could have been easily conceived of by a person skilled in the art based on the description in cited documents 1 and 2.

The inventions in claims 2-29 of the subject application should not be patented under Patent Law Section 29(2) since it could easily have been made by a person skilled in the art based on the description in cited documents 1 and 2.

The list of cited documents, etc.

1. WO 03/040948

See column 8, lines 4-23 and column 12, lines 1-9.

2. JP 06-222936 A

See paragraphs 0022-0023, 0025, 0030-0031 and 0044.

If any reason(s) for refusal is found later, the applicant will be notified accordingly.

(General instructions for filing amendments)

(1) When an amendment is made to the specification, the description changed as a result of the amendment shall be identified by an underline (Regulations under Patent Law Form 13 Remark 6).

(2) Matters that can be added in the amendment are matters described in the specification or drawings originally attached to the subject application or matters which are unambiguously derivable from the matters described in the specification or drawings as originally filed. The only amendments permissible are those directed to limited restriction of a claim, clarification of an ambiguous description or correction of an error in the description. It is advised that an

amendment be accompanied by an opinion stating for each amended matter how the amendment is legally valid and clearly pointing to a description in the specification as originally filed that supports the amendment (please refer to the format of demand for correction prepared in a trial for invalidation).



整理番号:SCEI02070 発送番号:034347 発送日:平成19年 1月30日 1

拒絶理由通知書

特許出願の番号	特願 2004-349195
起案日	平成19年 1月22日
特許庁審査官	殿川 雅也 9646 5B00
特許出願人代理人	森下 賢樹 様
適用条文	第29条第2項

この出願は、次の理由によって拒絶をすべきものである。これについて意見があれば、この通知書の発送の日から3か月以内に意見書を提出して下さい。

理 由

この出願の下記の請求項に係る発明は、その出願前に日本国内又は外国において、頒布された下記の刊行物に記載された発明又は電気通信回線を通じて公衆に利用可能となった発明に基いて、その出願前にその発明の属する技術の分野における通常の知識を有する者が容易に発明をすることができたものであるから、特許法第29条第2項の規定により特許を受けることができない。

記 (引用文献等については引用文献等一覧参照)

- ・請求項 1-29
- ・引用文献等 1-2
- ・備考

引用文献1には、キャッシュメモリを備えた複数のプロセッサを有する共有メモリ型マルチプロセッサ構成を有するコンピュータであって、スピンウェイト命令の実行中を検出した際には、プロセッサの動作モードを低消費電力モードに変更する、若しくは、バリア同期をとるために、他のプロセスの実行を行う（「他の処理タスクを開始する」）発明が記載されており、さらに、キャッシュステートを共有Sから無効Iに更新すると、この状態変化を値変更検出部で検出することにより、プロセッサに値変更通知を行う点が記載されている。

この値変更検出部による状態変化を安全に検出するために、予約付きロード命令及び予約時書き込み命令（「予約消失の通知」）を用いる技術が引用文献2に記載されており、本願の請求項1に係る発明は、引用文献1及び引用文献2の記載に基づいて当業者が容易に発明することができたものである。

同様に、本願の請求項2乃至29に係る発明は、引用文献1及び引用文献2の

整理番号:SCEI02070 発送番号:034347 発送日:平成19年 1月30日 2

記載に基づいて当業者が容易に発明することができたものであるから、特許法第29条第2項の規定により特許を受けることができない。

引用文献等一覧

1. 国際公開第03/040948号

第8欄第4行乃至同欄第23行、第12欄第1号乃至同欄第9行参照。

2. 特開平06-222936号公報

段落0022-0023, 0025, 0030-0031, 0044参照。

拒絶の理由が新たに発見された場合には拒絶の理由が通知される。

<補正に関する一般的な注意事項>

(1) 明細書を補正した場合は、補正により記載を変更した個所に下線を引くこと(特許法施行規則様式第13備考6)。

(2) 補正で付加できる事項は、この出願の出願当初の明細書又は図面に記載した事項のほか、出願当初の明細書又は図面の記載から自明な事項で行わなければならない。補正の際には、意見書で、各補正事項について補正が適法なものである理由を、根拠となる出願当初の明細書等の記載箇所を明確に示したうえで主張されたい。(意見書の記載は、無効審判における訂正請求書の記載形式を参考にされたい。)

先行技術文献調査結果の記録

・調査した分野 IPC G06F 9/46 - 9/54

・先行技術文献 米国特許第6275907号明細書

KAHLE, J. A., et al., Introduction to the Cell multiprocessor, IBM J. Res. & DEV., IBM, [online], 2005年 9月 7日, VOL. 49, No. 4/5, pp. 588 - 604 [検索日:平成19年1月22日]
<http://www.research.ibm.com/journal/rd/494/kahle.pdf>

この先行技術文献調査結果の記録は、拒絶理由を構成するものではない。

この拒絶理由通知の内容に関するお問い合わせがございましたら下記までご連絡下さい。

特許審査第四部情報処理 殿川雅也

整理番号:SCEI02070 発送番号:034347 発送日:平成19年 1月30日 3/E

TEL. 03 (3581) 1101 内線 3544